Investigation of Multilayer Perceptron and Class Imbalance Problems for Credit Rating

Zongyuan Zhao, Shuxiang Xu, Byeong Ho Kang
Mir Md Jahangir Kabir
School of Computing and Information Systems
University of Tasmania
Tasmania, Australia

Yunling Liu
College of Information and Electrical Engineering,
China Agricultural University
Beijing, China
Email: lyunling {at} 163.com

Abstract— **Multilayer perceptron (MLP) neural network is widely used in automatic credit scoring systems with high accuracies and efficiencies. However, class imbalance problems severely harm the prediction accuracy, when the number of instances in one class greatly overweighs the other class. In credit scoring datasets, class imbalance problems exist in fault detection models since there are always less unqualified cases than approved applications. In this work, we investigate the affection of different MLP structure to the prediction ability and develop a novel instance selection method to solve class imbalance problems in German credit datasets. We train 34 models 20 times with different initial weights and training instances. Each model has 6 to 39 hidden units in one hidden layer. Our test results prove that the prediction accuracy of the optimized model with our new instance selection methods is 5% higher than the best result reported in the relevant literature of recent years. We also summarize the tendency of scoring accuracy when the numbers of hidden units in MLP increases. The results of this work can be applied not only for credit scoring, but also in other MLP neural network applications, especially when the distribution of instances in a dataset is imbalanced.**

Keywords: credit scoring; neural network; German credit; class imbalance

I. INTRODUCTION

Credit rating has shown the ability to decrease credit risks and reduce "bad" loans, and has been widely used in banks and other financial institutes[1]. It is a set of decision models and their underlying techniques that help lenders judge whether an application of credit should be approved or rejected [2]. Basically credit scoring system can be mainly divided into two kinds: new credit application judgment and prediction of bankrupt after lending. The first kind uses personal information and financial status of a loan applicant as inputs to calculate a score. If the score is higher than a "safe level", the applicant has high possibility to preform good credit behavior. On the contrary, a low score means high risk for the loan so the lender needs to take careful consideration of the application. The other 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 the application scoring.

In recent years, artificial neural networks (ANN) has shown its advantages in credit scoring in comparison with linear probability models, discriminant analysis and other statistical techniques [3]. Compared with traditional credit scoring which is achieved by professional bank managers, automatic scoring has some obvious advantages: it saves costs and time for evaluating new credit applications; it is consistent and objective [4].

As a kind of ANN model, Multilayer perceptron (MLP) models have been widely utilized[2, 5, 6] which perform competitive prediction ability against other methods [7, 8]. In [9] back-propagation (BP) algorithm was developed and now has been widely used in training MLP feed-forward neural networks. Memetic pareto artificial neural network (MPANN) optimized BP algorithm using a multi-objective evolutionary algorithm and a gradient based local search [10]. This training method could reduce training time and at the same time enhance classification accuracy. The paper also presented a self-adaptive version called SPANN, which was obviously 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 [11]. But at the same time, some linear classifiers such as LDA and LOG also generated good results. This indicated that the performance differences between some models were not obvious [12]. Another test got similar results by testing the accuracy of several automatic scoring models using the German, Australian and Japanese credit datasets [13]. It reported that comparing with BP, C4.5 decision tree performed a little better for credit scoring but both of them could achieve high accuracies. Also, Nearest Neighbor and Naïve Bayes classifiers appeared to be the worst in their tests.

Improvements of 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 datasets [14]. They concluded that the learning scheme with 400 cases for training and 600 for validation performed best with an overall accuracy rate of 83.6%. Emotional neural network [15] 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 showed that both models were effective, but the emotional models outperformed the conventional ones in decision making speed and accuracy [16]. Another enhancement was artificial metaplasticity MLP, which was 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[17]. Fuzzy numbers can replace crisp weights and biases to overcome uncertainties and complexities in financial datasets [18]. Results showed that this hybrid classification model outperformed the traditional ANNs and also better than SVM, KNN (k-Nearest Neighbors) and others.

Hybrid systems which take neural networks as part of a whole construction and the combination system are researched in recent years. In [19], it presented a two-stage hybrid modelling procedure with ANN and multivariate adaptive regression splines (MARS). After using MARS in building the credit scoring model, the obtained significant variables then served as the input nodes of ANN. However, the improvements were not obvious. In fact, ensemble system performed better only in one of the three datasets in the experiments of [20]. The authors compared MLP with multiple classifiers or classifier ensembles, concluding that the ability of hybrid system was not better than usual methods and needed to consider all of them when making the optional financial decisions.

Support vector machine (SVM) and genetic algorithm (GA) are also used for credit rating with good performance. In [21] SVM model was refined by reduction of features using 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 [22], they selected important variables by GA to combine bank's internal behavioral 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 mortgage portfolio. Other than SVM and GA, Clustering-Launched Classification (CLC) is also available and may perform better than SVM [23]. A multi-criteria quadratic programming (MCQP) model was proposed based on the idea of maximizing external distance between groups and minimizing internal distances within a certain group [24]. It 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. Comparing with SVM it seemed more accurate and scalable to massive problems. Decision tree (DT) is another good alternative method. A tow dual strategy ensemble trees was developed based on bagging and random subspace [25]. This DT model reduced influences of noise data and redundant attributes of data to get relatively higher classification accuracy.

The class imbalance problem can appear in datasets where one class comprises considerably more samples than the other. It is a challenge to machine learning and data mining. In [26], 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, dynamic classifier and dynamic ensemble selection of features were added in the scoring model, which performed better than ensemble static classifiers [27].

In a credit scoring context, imbalanced data sets frequently occur as the number of defaulting loans in a portfolio is usually much lower than the number of observations that do not default [26]. Thus, it is very important to solve this problem when training credit scoring models. In [28] an improved over sampling approach based on synthetic minority over-sampling technique (SMOTE) was utilized before the training of credit rating models. Other experimental results using imbalanced credit datasets demonstrated that the use of resampling methods consistently improves the performance given by the original imbalanced data [29]. Besides, it is also important to note that in general, over-sampling techniques perform better than any under-sampling approach [30].

Other methods that can solve class imbalance problems include under sampling, feature selection, optimization of model structure and learning algorithms. Some of them have been tested on credit datasets and received good results while performances of others are still needed to be observed.

In order to improve the performance of credit scoring models, three methods will be discussed in this paper:

1. Develop a novel instance selection algorithm to deal with class imbalance problems. 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. By this method, the rate of different class data stays the same in training as the global rate. But at the same time they are chosen randomly, which also assumes fairness. We compare the accuracies of models made by this method with those from models with random choosing method.

2. Test the effect of different ratios of training-validation-testing data. [14] tested different ratios of training-validation data and obtained the best ratio by choosing the one with the highest accuracy. But in our model, there are three kinds of datasets: training, validation and test. 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 network. We train 34 models with the number of hidden neurons set to a number between 6 to 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 are 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: 1, we use instance selection to optimize instances used in training, validation and testing. Our new method can guarantee fairness and suit imbalance datasets. 2, we compare 34 models with different number of hidden units, to obtain the model with the highest accuracy and efficiency. Additionally, we have also explored 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 new instance selection methods and the MLP network models. Section 4 presents the experimental results and compares the accuracies of different models and methods. Finally Section 5 summarizes this work and gives directions for possible future work.

II. GERMAN CREDIT DATASET

The German credit dataset is a real world dataset with 21 features including 20 attributes recording personal information and financial history of applicants, which is available publicly on the UCI Machine Learning Repository website[13]. 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 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 utilized on this dataset which demonstrates 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 one. 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 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 [17] using 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.

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 (%)
[17]	MLP	84.67±1.5
[27]	Ensemble	82.03
[26]	LS-SVM	81.9
[18]	MLP	81.3
[16]	MLP	81.03
[21]	SVM	80.42
[25]	DT	78.52
[31]	Re-Rx	78.47
[32]	SVM	78.46
[33]	Case-based reasoning model	77.4
[34]	SVM	76.6
[35]	SVM	75.4
[30]	SVM	71.8

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 work to improve this classification accuracy rate.

III. MLP MODEL AND INSTANCE SELECTION

A. Instance Selection for Class Imbalance Problem

The class imbalance problems usually exist in credit datasets. However, modern classifiers assume that unseen data points on which the classifier will be asked to make a prediction are drawn from the same distribution as the training data [36]. 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 real world applications, as the input unseen data will be imbalanced, it is reasonable to keep the same ratio in the training and test dataset.

Another problem of data processing is the ratio of training-validation-test sets. All three sets should have proper

amount of instances. Usually, more instances for training can lead to better chance for getting a better model. However, as the amount of data is limited, more data used for training means less for validation and test. This will cause bad or unequal test performance. It is important to choose the best ratio since it 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 amounts of good and bad applications are

Good applications: $p * n$,
Bad applicaions: $(1 - p) * n$

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

Training data: $n * t$
Validation data: $n * v$
Test data: $n * (1 - t - v)$

As we want the ratio of good to bad applications stays the same in training data (as in original data), the training, validation and test data can be divided into good cases and bad cases. Table 3 shows the amount of each group and 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 minimize the effect of unordinary instances. This way, it is more likely to get a good data distribution which promotes a good scoring model.

By this method, all groups (training, validation and test) have the same ratio of good to bad instances. Traditional 10-fold validation method divides a dataset into 10 blocks of data randomly before training starts. It is not really random choosing and not fit for imbalanced dataset. 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 datasets of training, validation and test 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.

B. MLP Credit Scoring Model

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, then we 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 multilayer perceptron neural network.

In this work we use back propagation (BP) algorithm to train the network. This means the weights are altered by feeding back the differences between output signals and desired output values. It uses gradient decent method to control the speed of training. The neuron activation function is a simplified hyperbolic tangent sigmoid function "tansig". It can be calculated as follows:

$$f(x) = \frac{2}{1+e^{-x}} - 1 \quad (1)$$

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	$n * (1 - p) * t$	$n * (1 - p) * v$	$n * (1 - p) * (1 - t - v)$

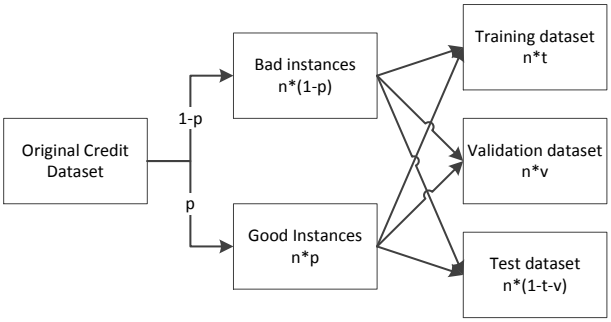

Figure 1 Flow of data processing and the amount of instances in each group

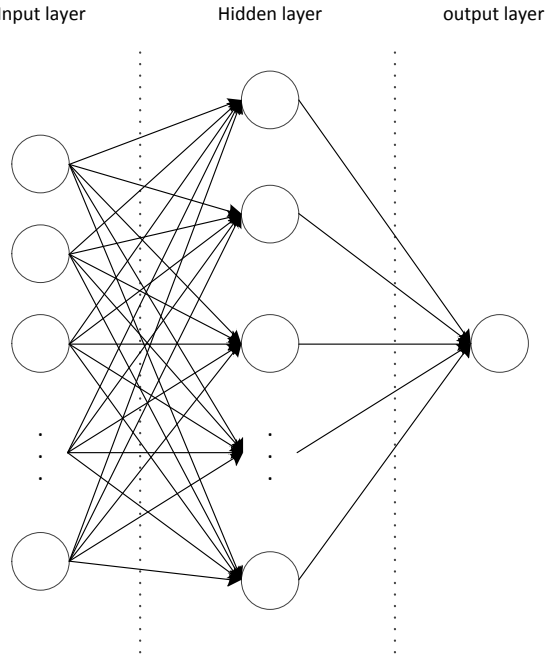

Figure 2 Typical structure of MLP neural network

Compared with sigmoid function, it is more cliffy around x=0. Experiment results denote that the tansig function can generate a larger gap on output values between different classes, while keeps high generalization abilities of MLP.

The number of hidden neurons can have 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 iteration goes on, the error rate of the network goes down until it reaches the lowest point. After that point, the error rate will not fall down any more and even rise. This is called over training or over fitting and the training procedure should be stopped when the error rate reaches the lowest point. Here we use some data to validate the network after every iteration. Validation data are not used to train the network, and they are 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.

IV. Experiments and Results

In this work we design experiments to find out a competitive MLP model for credit scoring. We use Matlab (version R2012a) software running on a 2.5GHz PC with 4GB RAM, Windows 7 OS. There are three aspects of the experiments. The first aspect is to find out the best amount of data used in training, validation and test, respectively. Then based on the results, the second aspect is to focus on our new instance choosing method. Thirdly, we discuss about 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 test are randomly chosen. The best and average error rates are listed and discussed.

A. *Choosing The Ratio Of Training-Validation-Test*

In the experiments, we use 3 different ratios of data, 800:100:100, 900:50:50 and 600:200:200 to find out 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. Also 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 result shows 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 which is also the best model of all. The average of 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 there is a low possibility to get a good model. This result will be used in the later experiments to simplify the training process.

B. *Training with Average Random Choosing Method*

In this section, we compare the models trained by 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 training, validation and test data groups. The ratio of training-validation-test is 800:200:200, which performs best in our previous tests. The results are listed on Table 4.

Comparing with the best and the average error rates in Table 4, the results of this round of tests are obviously 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 improves by 25% (the previous best rate is 0.208, as seen in Table 4). This indicates that, with this kind of data, it is more likely to get a high accuracy model which can have high predict ability.

Test results show that better organized 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 generalized to handling other credit datasets.

C. *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. As to the error rates in tests, the lowest ones seem to have been achieved when the number of hidden neuron is 9, or 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 it 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. But the average of all models is only 0.223, which means that there is little difference between models when only accuracy is considered. So 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 out in most cases (Shown in table 5). So 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 of validation is proper to reflect the prediction ability. It is always higher than 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. It is 0.05 higher than the best result with test data. Although this value is not as objective as the accuracy from pure test data, it indicates that MLP model has the ability of rating credit application more precisely. Also, there is an interesting tendency found in the validation dataset. As the number of hidden units gets larger, this error rates seems to be lower. It is shown in Fig. 3.

Table 4 Test results and comparison of different ratios of data, and two instance choosing methods.

	800:100:100		**900:50:50**		**600:200:200**		**Average Random Choosing method**			
Number of hidden neurons	*Lowest error rate*	*Average error rate*	*Lowest error rate*	*Average error rate*	*Lowest error rate*	*Average error rate*	*Lowest test error rate*	*Average error rate*	*Lowest validate error rate*	*Average validate error rate*
6	0.21	0.2415	0.21	0.2925	0.245	0.273	0.14	0.2145	0.1265	0.1502
7	0.21	0.251	0.2	0.251	0.225	0.273	0.15	0.211	0.1228	0.1441
8	0.19	0.239	0.21	0.2765	0.22	0.2615	0.17	0.2215	0.1231	0.1438
9	0.18	0.2385	0.18	0.2315	0.245	0.27675	0.13	0.2085	0.1284	0.1472
10	0.18	0.2295	0.2	0.243	0.24	0.27075	0.13	0.221	0.1270	0.1471
11	0.2	0.242	0.18	0.2635	0.225	0.26275	0.16	0.223	0.1234	0.1439
12	0.21	0.246	0.19	0.2675	0.23	0.26575	0.13	0.225	0.1297	0.1491
13	0.2	0.256	0.21	0.2705	0.2	0.26625	0.16	0.2225	0.1163	0.1431
14	0.21	0.258	0.19	0.258	0.205	0.26525	0.14	0.2105	0.1113	0.1407
15	0.17	0.2595	0.21	0.26	0.24	0.2715	0.16	0.219	0.1103	0.1422
16	0.22	0.246	0.23	0.26	0.23	0.266	0.14	0.2085	0.1245	0.1399
17	0.19	0.248	0.21	0.245	0.235	0.2745	0.19	0.2415	0.1196	0.1433
18	0.21	0.2595	0.21	0.293	0.21	0.26725	0.17	0.221	0.1022	0.1342
19	0.2	0.255	0.24	0.277	0.235	0.28	0.14	0.2135	0.1124	0.1407
20	0.21	0.2585	0.21	0.262	0.22	0.2655	0.16	0.2215	0.1006	0.1402
21	0.22	0.256	0.21	0.2835	0.24	0.2665	0.16	0.215	0.1207	0.1368
22	0.2	0.2545	0.22	0.2985	0.23	0.26475	0.15	0.2195	0.1217	0.1351
23	0.19	0.259	0.21	0.2675	0.24	0.27075	0.18	0.2265	0.1143	0.1353
24	0.19	0.26	0.2	0.26	0.245	0.27375	0.14	0.215	0.1249	0.1377
25	0.22	0.257	0.21	0.2845	0.24	0.27175	0.15	0.223	0.1007	0.1394
26	0.22	0.2725	0.19	0.276	0.235	0.28575	0.14	0.224	0.0884	0.1310
27	0.2	0.261	0.22	0.287	0.24	0.2735	0.17	0.2175	0.1227	0.1380
28	0.21	0.2625	0.23	0.2755	0.235	0.2755	0.15	0.2235	0.1123	0.1345
29	0.21	0.262	0.19	0.266	0.225	0.273	0.15	0.2265	0.1100	0.1339
30	0.23	0.2765	0.23	0.278	0.21	0.2665	0.15	0.2275	0.0916	0.1304
31	0.24	0.2715	0.23	0.2775	0.25	0.28575	0.18	0.228	0.0892	0.1283
32	0.22	0.265	0.2	0.2775	0.245	0.272	0.18	0.234	0.1013	0.1295
33	0.23	0.2755	0.22	0.273	0.245	0.27725	0.17	0.2345	0.1070	0.1287
34	0.22	0.272	0.22	0.281	0.24	0.28575	0.17	0.228	0.1023	0.1285
35	0.22	0.2675	0.22	0.2885	0.235	0.28225	0.14	0.229	0.0932	0.1243
36	0.24	0.2835	0.19	0.285	0.245	0.2895	0.14	0.2275	0.1055	0.1311
37	0.22	0.278	0.23	0.3175	0.225	0.29125	0.16	0.2265	0.0963	0.1258
38	0.17	0.28	0.22	0.301	0.21	0.27825	0.19	0.2385	0.0801	0.1324
39	0.24	0.294	0.2	0.284	0.235	0.284	0.14	0.2375	0.0854	0.1215
Best	0.17	0.2295	0.18	0.2315	0.2	0.2615	0.13	0.2085	0.0801	0.1215
Average	0.20823	0.25988	0.20941	0.27389	0.23161	0.27375	0.1553	0.2231	0.1102	0.1368

Table 5 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

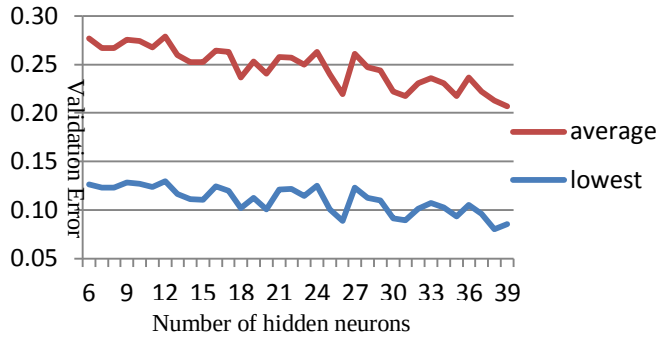

Figure 3 Tendency of model's error rates when numbers of hidden units increase

D. *Summary of Experiments*

We compare the accuracies of models trained with different ratios of training-validation-test data at first. The 800:100:100 group 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 optimize the dataset. The ratio of approved/rejected instances in 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 among all models, in consideration of high accuracy and low computational time. Also we find an interesting relationship between number of hidden neurons and validation accuracy: the more hidden units a model has, the higher 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.

V. CONCLUSIONS

In this paper, we present a comparison of MLP models trained by BP algorithm with different inputs and numbers of hidden units. We also introduce an effective new method for choosing instances for training, validation and test datasets. Our new method is called Average Random Choosing. This method has been proved to be effective for increasing the accuracy of a model, especially when the original dataset is imbalanced between its classes of data. In our approach we have trained 34 kinds of models with numbers of hidden units ranging from 6 to 39. For each kind we have 20 models initialized with random weights to avoid accidental low accuracy. We also test different ratios of training-validation-test to choose the most suitable amount of instances for each dataset.

In our experiments, we use a real world dataset (the German dataset which has been frequently used in previous works). We summarize the results of exiting works related to this dataset. Then we design the structure of MLP neural networks with the BP training algorithm, and compare our results against them. Validation is added into training to avoid over fitting. During the experiments, we record and analyses the error rates of validation and test datasets.

In conclusion, we get an acceptable model with higher accuracy for credit rating. This MLP model contains 9 hidden units, trained by 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 test, which is higher by 5% than the best results of existing works related to the German dataset. We also believe our method can be extended to other credit datasets. In the future, our work would focus on reducing attributes (feature or attribute selection) which may significantly save computation time for training an MLP for this and other similar applications.

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

[1] Thomas, L.C., D.B. Edelman, and J.N. Crook, Credit Scoring and Its Applications. 2002: SIAM: Philadelphia, PA.

[2] Jensen, H.L., Using Neural Networks for Credit Scoring. Managerial Finance, 1992. 18(6): p. 15.

[3] Marques, A.I., V. Garcia, and J.S. Sanchez, A literature review on the application of evolutionary computing to credit scoring. Journal of the Operational Research Society, 2013. 64(9): p. 1384-1399.

[4] Saberi, M., et al., A granular computing-based approach to credit scoring modeling. Neurocomputing, 2013. 122: p. 100-115.

[5] Zhong, H., et al., Comparing the learning effectiveness of BP, ELM, I-ELM, and SVM for corporate credit ratings. Neurocomputing, 2014. 128(0): p. 285-295.

[6] Oreski, S., D. Oreski, and G. Oreski, Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment. Expert Systems with Applications, 2012. 39(16): p. 12605-12617.

[7] West, D., Neural network credit scoring models. COMPUTERS & OPERATIONS RESEARCH, 2000. 27(11-12): p. 1131-1152.

[8] Šušteršič, M., D. Mramor, and J. Zupan, Consumer credit scoring models with limited data. Expert Systems with Applications, 2009. 36(3): p. 4736-4744.

[9] Werbos, P.J., BEYOND REGRESSION: NEW TOOLS FOR PREDICTION AND ANALYSIS IN THE BEHAVIORAL SCIENCES, in Harvard University1975, Harvard University.

[10] Abbass, H.A., Speeding Up Backpropagation Using Multiobjective Evolutionary Algorithms. Neural Computation, 2003. 15(11): p. 2705-2726.

[11] Baesens, B., et al., Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring. The Journal of the Operational Research Society, 2003. 54(6): p. 627-635.

[12] Marqués, A.I., V. García, and J.S. Sánchez, Exploring the behaviour of base classifiers in credit scoring ensembles. Expert Systems with Applications, 2012. 39(11): p. 10244-10250.

[13] Bache, K. and M. Lichman. {UCI} Machine Learning Repository. 2013; Available from: http://archive.ics.uci.edu/ml.

[14] Khashman, A., Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. Expert Systems with Applications, 2010. 37(9): p. 6233-6239.

[15] Khashman, A., A Modified Backpropagation Learning Algorithm With Added Emotional Coefficients. Neural Networks, IEEE Transactions on, 2008. 19(11): p. 1896-1909.

[16] Khashman, A., Credit risk evaluation using neural networks: Emotional versus conventional models. Applied Soft Computing, 2011. 11(8): p. 5477-5484.

[17] Marcano-Cedeño, A., et al., Artificial metaplasticity neural network applied to credit scoring. International Journal of Neural Systems, 2011. 21(4): p. 311-317.

[18] Khashei, M., et al., A bi-level neural-based fuzzy classification approach for credit scoring problems. Complexity, 2013. 18(6): p. 46-57.

[19] Lee, T.-S. and I.F. Chen, A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. Expert Systems with Applications, 2005. 28(4): p. 743-752.

[20] Tsai, C.-F. and J.-W. Wu, Using neural network ensembles for bankruptcy prediction and credit scoring. Expert Systems with Applications, 2008. 34(4): p. 2639-2649.

[21] Hens, A.B. and M.K. Tiwari, Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method. Expert Systems with Applications, 2012. 39(8): p. 6774-6781.

[22] Chi, B.-W. and C.-C. Hsu, A hybrid approach to integrate genetic algorithm into dual scoring model in enhancing the performance of credit scoring model. Expert Systems with Applications, 2012. 39(3): p. 2650-2661.

[23] Luo, S.-T., B.-W. Cheng, and C.-H. Hsieh, Prediction model building with clustering-launched classification and support vector machines in credit scoring. Expert Systems with Applications, 2009. 36(4): p. 7562-7566.

[24] Peng, Y., et al., A Multi-criteria Convex Quadratic Programming model for credit data analysis. Decision Support Systems, 2008. 44(4): p. 1016-1030.

[25] Wang, G., et al., Two credit scoring models based on dual strategy ensemble trees. Knowledge-Based Systems, 2012. 26(0): p. 61-68.

[26] Brown, I. and C. Mues, An experimental comparison of classification algorithms for imbalanced credit scoring data sets. Expert Systems with Applications, 2012. 39(3): p. 3446-3453.

[27] Xiao, J., et al., Dynamic classifier ensemble model for customer classification with imbalanced class distribution. Expert Systems with Applications, 2012. 39(3): p. 3668-3675.

[28] Li, Z. and W. WenXian, A Re-sampling Method for Class Imbalance Learning with Credit Data. Proceedings of the 2011 International Conference on Information Technology, Computer Engineering and Management Sciences (ICM 2011), 2011: p. 393-397.

[29] Garcia, V., A.I. Marques, and J.S. Sanchez, Improving risk predictions by preprocessing imbalanced credit data. Neural Information Processing. 19th International Conference (ICONIP 2012). Proceedings, 2012: p. 68-75.

[30] Marques, A.I., V. Garcia, and J.S. Sanchez, On the suitability of resampling techniques for the class imbalance problem in credit scoring. Journal of the Operational Research Society, 2013. 64(7): p. 1060-1070.

[31] Setiono, R., B. Baesens, and C. Mues, RULE EXTRACTION FROM MINIMAL NEURAL NETWORKS FOR CREDIT CARD SCREENING. International Journal of Neural Systems, 2011. 21(4): p. 265-276.

[32] Yu, L., et al., Credit risk evaluation using a weighted least squares SVM classifier with design of experiment for parameter selection. Expert Systems with Applications, 2011. 38(12): p. 15392-15399.

[33] Vukovic, S., et al., A case-based reasoning model that uses preference theory functions for credit scoring. Expert Systems with Applications, 2012. 39(9): p. 8389-8395.

[34] Ping, Y. and L. Yongheng, Neighborhood rough set and SVM based hybrid credit scoring classifier. Expert Systems with Applications, 2011. 38(9): p. 11300-11304.

[35] Gonen, G.B., M. Gonen, and F. Gurgen, Probabilistic and discriminative group-wise feature selection methods for credit risk analysis. Expert Systems with Applications, 2012. 39(14): p. 11709-11717.

[36] Wasikowski, M. and C. Xue-wen, Combating the Small Sample Class Imbalance Problem Using Feature Selection. Knowledge and Data Engineering, IEEE Transactions on, 2010. 22(10): p. 1388-1400.