Workshop on Artificial Intelligence

Assessing Check Credit With Skewed Data: A Knowledge Discovery Case Study

Chun-Min Hung, Yueh-Min Huang, Tse-Sheng Chen

Department of Engineering Science, National Cheng Kung University, Taiwan, R.O.C.

Email: goodmans@giga.net.tw, {Raymond, tsch}@mail.ncku.edu.tw

Tel. 06-2757575-63342-30, 06-2757575-63336, 06-2757575-63322 Fax 06-2766549

Contact author: **Chun-Min Hung**

Abstract

There have been lots of studies focusing on the improvement of performance of the whole classification in data mining, but few of them concern the relatively accuracy rate within subclasses of the whole classification. However, a real world data may exist many unproportionate subsets such that the supervised learning misleads a model into that of high performance but low interesting. In this work, we attempt to predict the customers' credit of a checking account in a bank with skewed data and strive to fairly increase the accuracy rate of multi-object classification using neural network. Our objective is to determine a range of credit scoring which could be granted to the manger under a critical credit risk. In this paper, we analyze a real case of banking data using a series of elaborate experiments on some famous algorithms of neural network and to discuss how to decide the parameters of classes with a skewed data during training process. The result shows that the accuracy rate of classification of all classes is about 79% on average without data clean and the accuracy rates of classification of subclasses increase up to 64% with data clean. Based on the observation of experimental data, the conclusion is that this overdraft's attribute dominates the average accuracy rate of classification while the proportion of training datasets govern the accuracy rate of classification of interesting subclasses. Finally, we infer that a nearly 15% range of noise about this real case is reasonable. Hence, a bank manager can be granted a range of 15 % for credit scoring on his duty.

Key words: data mining, skewed data, neural network, bank

Assessing Check Credit With Skewed Data: A Knowledge Discovery Case Study

Chun-Min Hung, Yueh-Min Huang, Tse-Sheng Chen

Department of Engineering Science, National Cheng Kung University, Taiwan, R.O.C.

goodmans@giga.net.tw, {Raymond, tsch}@mail.ncku.edu.tw

Abstract

There have been lots of studies focusing on the improvement of performance of the whole classification in data mining, but few of them concern the relatively accuracy rate within subclasses of the whole classification. However, a real world data may exist many unproportionate subsets such that the supervised learning misleads a model into that of high performance but low interesting. In this work, we attempt to predict the customers' credit of a checking account in a bank with skewed data and strive to fairly increase the accuracy rate of multi-object classification using neural network. Our objective is to determine a range of credit scoring which could be granted to the manger under a critical credit risk. In this paper, we analyze a real case of banking data using a series of elaborate experiments on some famous algorithms of neural network and to discuss how to decide the parameters of classes with a skewed data during training process. The result shows that the accuracy rate of classification of all classes is about 79% on average without data clean and the accuracy rates of classification of subclasses increase up to 64% with data clean. Based on the observation of experimental data, the conclusion is that this overdraft's attribute dominates the average accuracy rate of classification while the proportion of training datasets govern the accuracy rate of classification of interesting subclasses. Finally, we infer that a nearly 15% range of noise about this real case is reasonable. Hence, a bank manager can be granted a range of 15 % for credit scoring on his duty.

Key words: data mining, skewed data, neural network, bank

1. Introduction

Nowadays, digital information is relatively easy to acquire and fairly inexpensive to store and many data mining approaches have been implemented to analyze it. The major research directions include the clustering, classification, association rules, and prediction in data mining, and the well-known methods employed are neural network (NN)[3], genetic algorithm, some primary data mining tools, and fuzzy logic. However, certain real world data already collected is far behind that generated. Especially, it is difficult to collect certain data under some authorization constraints. Fortunately, we collect a certain of encoded dataset by a cooperating project from a local bank TNB (Tainan Business Bank). Some literatures of real case studies can be found on [1][2].

Problem

Most field studied have worked out on the improvement of performance of the whole classification in data mining. Although they emphasize on different viewpoints in vary methods, it still lacks of the discussion of this relatively accuracy rate within subclasses of the whole classification. Hence, a classification of the special subclass whether may gain a relatively fair accuracy rate of classification against others become an interesting problem. Especially, a real world data actually exists many unproportionate subsets such that the supervised learning misleads a model into that of high performance but low interesting. On the other part, the credit scoring plays a more important role under slack economy. How to assess a customers credit to for the loan becomes a critical work in a bank. Furthermore, the credit of checking account of customers is one of primary crediting inspections. Therefore, we focus on the problem of the classification that classifies the datasets into three classes: good, risk, and declination respectively. In particular, this risk class is interesting to analyze because it is a difficult task for pattern recognition by human decision. Finally, we need to know which attributes dominate the accuracy rate of classification and what factors dominate the accuracy rate of classification of interesting subclasses, especially for risk classes.

In the problem of classification or prediction, we use the techniques of neural network as a tool to analyze our datasets. A multiplayer feedforward networks is an important class of neural networks [1]. In this work, we employ over ten kinds of algorithm to make an adaptive learning model in order to predict this credit scoring of checking account for a local bank in Taiwan. The work has collected about sixty thousands records as inputs. For efficiency, we only retrieval a part of original data source to cross-analyze the complex combination of environment parameters of neural network. In fact, it is enough to simulate behavior of original data source by means of a simple test.

In this study, the following requirements are necessary to approach our objective:

I. We need to know what suitable parameters of neural network used for training in the real case. It includes a learning rate, the number of epoch, mean square error, the properties of algorithm, the number of neurons, and the analysis on execute time.

II. The constructed model must be tested its robustness. It includes outside testing, overfitting problem [4], and noising handling.

III. For special purpose, the risk class which given the smaller datasets than others need to improve the accuracy rate of classification up to a relatively fair proportion against others.

The experimental processes need obey the basic step on the knowledge discovery process. It includes data clean, PCA selection, pre-processing, data mining, and interpretation [5] and the remove of a variable for finding out dominated attributes.

2. Data Mining

2.1. The Knowledge Discovery Process

The process of knowledge discovery via data mining can generally be divided into four basic activities: selection, pre-processing, data mining, and interpretation [5]. Figure 1 shows that the knowledge discovery process in this real case. The former item of parentheses is an abbreviation of the techniques we used in that stage and the latter item is a major objective of task we assigned in that current stage. First, we use a Structure

Query Language (SQL) tool to select a part of source data we want from the bank's database server. And, we clean these source data called target datasets through a series of pre-processing which contains certain manual and automatic processes including a technique of principal component analysis (PCA) [3]. Furthermore, we employ the product of MATLAB (MATrix LABoratory) Neural Network Toolbox 4.0 of MathWorks® Corp. to accomplish our training exercises.

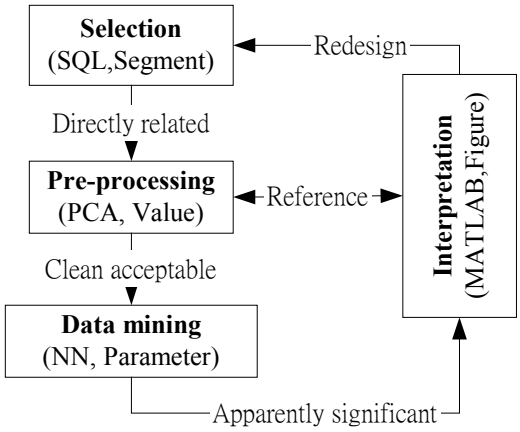

Figure 1. The knowledge discovery process

For selection, we create a target dataset of about 1,000 records from the part of original dataset 6,2621 records. The number of target records is reasonable due to nearly 1000 records via checking counts opened per year in this bank. Because of the credit environment changed and the business rules transformed by the policy of company year by year, the complete dataset should not be submitted to the neural networks tool that is one of data mining software used in our work. Again, we focus the effort to find out the pattern of special subclass so that any inconsistencies should be avoided during that periodic time.

In pre-processing, we inspect the schema of which is a configuration of datasets on the database and determine to discard about 20 fields which include the irrelevant fields, private fields, missing data and to only retain six fields that contain 'The type of overdraft', 'A type of established account', 'The count of transactions', 'The status of recently transactions', 'An interest's contributions this year', 'The signature of specimen seal whether has been changed'. For abbreviation, we marked them from variable 1 to 6 on certain figures, respectively. In addition, a principal component analyze is used to automatically choice primary attributes by computer. Notes that those retained fields might contain skewed fields and outlying data points, this is the aim we like to attack. Finally, we convert the data into the format acceptable by the neural network. Further cleaning will be described in experimental section.

For data mining, we choose a tool of neural networks as data mining software and particularly take the type of multiplayer feed-forward networks [3] that contain many algorithms analyzed in our experiments. In the

experiments, the R-coefficient [7] is used to measure how close between output variable and target variable. It's value closes to 1 that is to say extremely fitting between output and target.

For interpretation, the analysis and interpretation of the results produced. The results of this stage part will be described in the experimental section.

2.2. Training exercise

A research project is provided by TNB with data mining training. As part of that training exercise, my colleagues and I performed a preliminary study with data extracted from the TNB data warehouse. The data, a small sample of current checking accounts, contains about 1,000 records, roughly 2 percent of the TNB's current database.

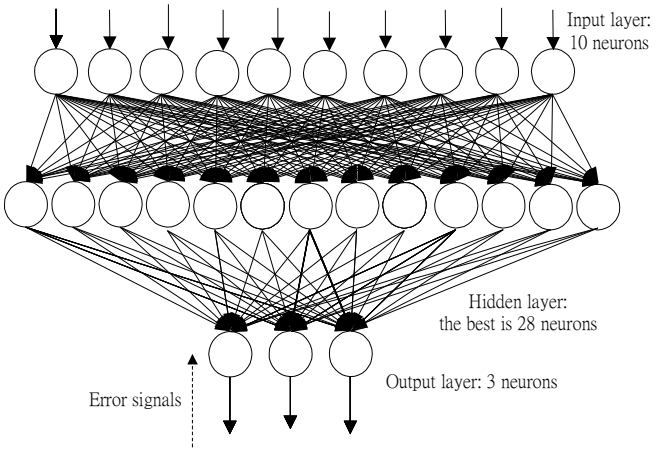

Figure 2.. Architectural graph of a mutilayer peron with one hidden layers

Before training exercise, input variables and out variable must be encoded into numeric form for the predictive modeling techniques using neural networks. We planned to construct a learning model through this information and extract the pattern of the risk subclass of all checking accounts and then suggest a remark of this descriptive modeling to the bank. Typically, the network shown as figure 2 consists of a set of neurons that constitute the input layer, one or more hidden layers of neurons, and an output layer of neurons. The input signal propagates through the networks are commonly referred to as multiplayer perceptrons (MLPs) [3].

In this real case, the six input variables are encoded into 10 neurons totally. Two neurons {0,1} of input variable '1' denote non-overdraft type. Two neurons {1,0} of input variable '1' denote vouching-overdraft type. Four neurons {0, 0,0,1} of input variable '2' denote chief-guarantee type. Four neurons {0,0,1,0} of input variable '2' denote sponsor-guarantee type. Four neurons {0,1,0,0} of input variable '2' denote high contribution type. Four neurons {1, 0,0,0} of input variable '2' denote others type. One neuron a range of [0,1]

of input variable ‘3’ denotes the count of transactions. One neuron of input variable ‘3’ which simply log mapped to a range of [0,1] denotes the count of transactions. One neuron of input variable ‘4’ which simply log mapped to a range of [0,1] denotes the contribution of interest this year, too. One neuron of input variable ‘5’, whose value 1 denotes that the customer ever executed transactions during recently three months, or its value is 0. Finally, one neuron of input variable ‘6’, whose value 1 denotes that the customer ever change of its signature of specimen seal, or its value is 0.

The output variable is a degree of credit, which is divided into three subclasses from all customers in checking accounts and is encoded into three neurons. Three neurons {1,0,0} of output variable denote ‘1’ class that is a group of declined customers, three neurons {0,1,0} of output variable denote ‘2’ class that is a group of risk customers, and three neurons {0,0,1} of output variable denote ‘3’ class that is a group of good customers. In this paper, the skew data means that a group of good customers are extremely larger than others.

Besides fixed parameters mentioned above, the experimental process will test many parameters’ combinations for the questions include the number of neurons of hidden layer, learning rate, algorithm, execute time, dominated attributes, noisy ratio, the proportion of training dataset of subclass, the effect of data clean and so forth. Of course, the aim of this exercise is to understand a predictive model of three subclasses, whether it is feasible or not. We assume that the prediction of outside testing is feasible and their accuracy rates of those subclasses are simultaneously over 50%.

3. Back-propagation learning

In this section, we will briefly introduce the famous ten algorithms of back-propagation learning [8] shown in Table 1 for preparing our experiments. This error back-propagation learning consists of two passes which one pass propagates the effect from the input layer through hidden layers to the output layer, and then another pass backward propagates an error signal through the network in light of a inverse order of the original pass. Within these neurons of hidden layers and output layers, a simulation of synaptic activity needs to assign a transfer function for the neural networks in the same layer. Here, we assign Tangent-Sigmoid function to be a transfer

function of the hidden layer and assign linear function to be a transfer function of the output layer. For convenient to represent, we list a set of notation of symbols shown in Table 2.

Table 1. The abbreviation of ten back-propagation algorithms

Abbreviation	The meanings of parameters
Traingd	Gradient Decent (Batch mode)
Traingda	Gradient Decent with adaptive learning rates
Traingdm	Gradient Decent with momentum
Traingdx	Gradient Decent with adaptive learning rates and momentum
Trainb	Batch training of learning rule via weight and bias
Trainbfg	quasi-Newton algorithm
Trainc	Incremental learning of a sorted cycle
Traincgb	Conjugate Gradient (Powell-Beale Restarts)
Traincgf	Conjugate Gradient (Fletcher-Reeves)
Trainlm	Levenberg-Marquardt algorithm

Table 2. The parameters of back-propagation algorithm

Symbol	The meanings of parameters
$\Re_{av}$	Average error energy function
$F(\mathbf{w},\mathbf{p}(n))$	Error vector of the n-th epoch
$\mathbf{e}$	Error vector
$\mathbf{t}$	Target output of networks
$\mathbf{a}$	Actual output of networks
$\mathbf{p}$	Input vector
$\mathbf{w}$	Weights and bias vector
$\mathbf{r}$	Gradient vector
$\mathbf{s}$	Conjugate vector
$\mathbf{H}$	Hessian matrix
$\mathbf{J}$	Jacobian matrix
α	Learning rate
β	Coefficient of conjugate gradient
μ	Coefficient of Levenberg-Marquardt algotithm

3.1. Gradient Steepest Descent

For a given training dataset, we define the average error energy $\Re_{av}$ as a function of the weights and bias denoted by vector **w.** The vector **w** is a set of parameters of neurons of this network. The objective of the learning process is to minimize $\Re_{av}$. Hence, we have

$$\Re_{\text{av}}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(t(n)-a(n))^2\ , \tag{1}$$

where N is the number of records of the entire training dataset.

Those methods compute the gradient by the first derivative of $\Re_{av}$ with respect to **w**. Therefore,

$$\mathbf{r}(n) = \frac{\partial \Re_{\text{av}}}{\partial \mathbf{w}} = -\frac{1}{N}\sum_{n=1}^{N}\frac{\partial F(\mathbf{w},\mathbf{p}(n))}{\partial \mathbf{w}}(t(n)-a(n))^2 \tag{2}$$

The best simple back-propagation algorithm is gradient steepest descent algorithm. By this proof of chain rule of Calculus, we are given a formula for the modification of weight and bias,

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \alpha(n)\mathbf{r}(n) \tag{3}$$

We use the transformations of gradient steepest descent methods to train and test our dataset such as 'Traingd', 'Traingda', 'Traingm', and 'Traingx' which their briefly are listed in Table 1. Moreover, we have two modes to implement the modification of weight and bias of the algorithms like 'Trainb' and 'Trainc'.

3.2. Conjugate Gradient

Mentioned methods adjust weight vector **w** in the deepest descent direction. These methods maybe descend error function quickly. It's not necessary to make convergence fast. The conjugate gradient method [10] belongs to a class of second-order optimization methods known collectively as conjugate-direction methods. This method can accelerate the typically slow rate of convergence experienced with the method of steepest descent and need to compute a Hessian matrix **H**(n) defines as follows:

$$\mathbf{H}(N) = \frac{\partial^2 \Re_{\text{av}}}{\partial \mathbf{w}^2} = \frac{1}{N}\sum_{n=1}^{N}\left\{\left(\frac{\partial F(\mathbf{w},\mathbf{p}(n))}{\partial \mathbf{w}}\right)\left(\frac{\partial F(\mathbf{w},\mathbf{p}(n))}{\partial \mathbf{w}}\right)^T - \frac{\partial^2 F(\mathbf{w},\mathbf{p}(n))}{\partial \mathbf{w}^2}(t(n)-a(n))\right\} \tag{4}$$

In conjugate gradient algorithm, it can avoid using a Hessian matrix **H**(n) which is plagued with computational difficulties, to determine the next direction of search is conjugated. Therefore, we have

$$\mathbf{s}(n) = \mathbf{r}(n) + \beta(n)\mathbf{s}(n-1), n = 1,2,...N-1\ , \tag{5}$$

where **s**(n) is a conjugate vector of direction of search.

Here, we choose two major ways to achieve the evaluation of a coefficient β(n) of conjugate gradient to determine the direction of search for a minimization of average error energy function $\Re_{av}$ without explicit knowledge of the Hessian matrix **H**(n). One way is Fletcher-Reeves formula [11], [12] as follows:

$$\beta(n) = \frac{\mathbf{r}^{T}(n)\mathbf{r}(n)}{\mathbf{r}^{T}(n\text{-}1)\mathbf{r}(n\text{-}1)}, \tag{6}$$

where $\mathbf{r}^{T}$ is a transport vector of a gradient vector **r**.

Another way is Powell-Beale Restarts formula [10] as follows:

$$| r^{T}(n-1)r(n) \geq 0.2 \,||\, r(n) \,||^{2}, \tag{7}$$

where 0.2 is the default parameter of an orthogonality ratio.

This formula means that the direction of search will periodically be reset to a negative gradient direction if certain tiny orthogonality existed. Fletcher-Reeves formula and Powell-Beale restarts methods are tested by our experiments denoted their abbreviation of function as 'Traincgf' and 'Traincgb', respectively.

Here, we introduce another selection for an optimization problem. That is called a Newton method, which its basic formula is

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mathbf{H}^{-1}(n)\mathbf{r}(n), \tag{8}$$

where $\mathbf{H}^{-1}(n)$ is the inverse Hessian matrix.

But it is impractical method because the computation of an inverse Hessian matrix $\mathbf{H}^{-1}$ is difficult. We therefore use a quasi version of Newton method and to be denoted by 'Trainbfg' in our experiments. It can avoid the computational requirements associated with the evaluation, storage, and inversion of the Hessian matrix in Newton's method. A quasi-Newton method only takes an estimate of the gradient vector **r** and use second-order information about the average error energy function to update the approximated Hessian matrix **H'** without actually taking knowledge of the Hessian matrix **H**. The method is assured of going downhill on the error surface [9].

3.3. Levenberg-Marquardt algorithm

Levenberg-Marquardt algorithm [13] is designed to a class of second-order optimization methods and is similar to a quasi-Newton method but without the computation of Hessian matrix. A Hessian matrix move in on the formula $\mathbf{H}=\mathbf{J}^{\mathrm{T}}\mathbf{J}$ and the gradient $\mathbf{r}=\mathbf{J}^{\mathrm{T}}\mathbf{e}$ if $\Re_{av}$ is formed by the summation of square, where **J** is Jacobian matrix which contains the first derivate of $\Re_{av}$ with respect to **w** and **e** is error vector of the network. We rewrite Equation **(8)** and combination above formula as follows:

$$\mathbf{w}(n+1)=\mathbf{w}(n)-[\mathbf{J}^{\mathrm{T}}\mathbf{J}+\mu\mathbf{I}]^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{e}\,, \tag{9}$$

where μ is a tuning parameter and **I** is an identity matrix.

This method indicates a two-ways form between the quasi-Newton method and the smaller gradient descent method via adaptively tuning the μ-value from zero to very large value while $\Re_{av}$ from decreasing to increasing. The method employs the vantages of which the quasi-Newton method is more quickly and correctly when it approaches to nearby minimum. However, the computation of Jacobian matrix **J** is more complex than that of Hessian Matrix **H** and the implementation of algorithm will also consume very large memory for storing Jacobian matrix **J** which contains Q x n sizes, where Q is the number of records of training dataset and the n is the dimension of weight vector **w**. Fortunately, the Jacobian matrix **J** can be decomposed into the following form:

$$\mathbf{H}=\mathbf{J}^{T}\mathbf{J}=\left[\mathbf{J}_1^{T}\mathbf{J}_2^{T}\right]\begin{bmatrix}\mathbf{J}_1\\ \mathbf{J}_2\end{bmatrix}=\mathbf{J}_1^{T}\mathbf{J}_1+\mathbf{J}_2^{T}\mathbf{J}_2 \tag{10}$$

By summation of sub-items, this decomposition of matrix can reduce the requirement of memory. In our experiments, we have not tested since the problem of memory is not so critical in this case.

4. Experimental Data

In this section, we will list a part of the experimental data for interpretation, including environment setup, basic experiments , noisy interference, data clean, and training proportionality.

4.1. Environment setup and basic experiments

In the environments, we have trained and tested the network under the PC's environment, which contains 512 MB RAM, 1.5G Pentium IV CPU, and the platform of Windows 2000.

Now, suppose we have selected our target dataset from the source database and have formatted the input vector as a numeric matrix that the tools needs. We would also analyze the performance of these networks with varied algorithms via measuring the mean square error in our experiment. In the very beginning, the first step of experiments is to decide the range of learning rates since we might have no knowledge of these analyzed datasets.

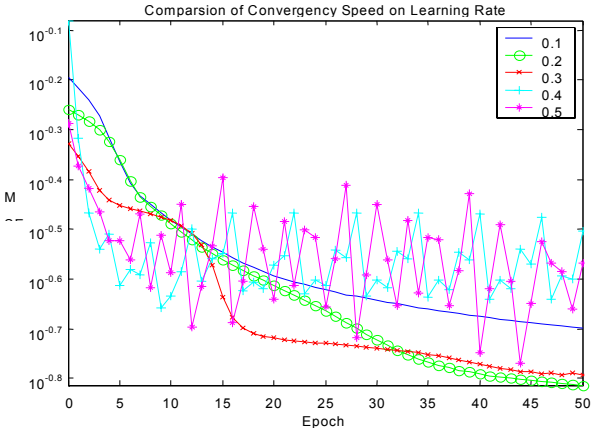

Figure 3. The comparison of a convergence speed on different learning rates, the y-axis is a logarithmic scale

Figure 3 shows that a value of nearly 0.2 is preferred to as initial learning and as benchmark with different algorithms. And it also shows that make a divergence of performance over a value of roughly 0.4. Hence, the following experiments would fix a value of 0.2 on all other training processes as their initial learning rate. The x-axis is Epoch means that the training processes repeat 50 times through the whole training dataset. The y-axis represents mean square error at each epoch.

Shifted to the next experiment, it is similar to the mentioned method. We would like to have the knowledge of how many hidden neurons are enough for most algorithms because that the number of hidden neurons affects the capacities of fitting the non-linear curves on those multi-dimensions variables.

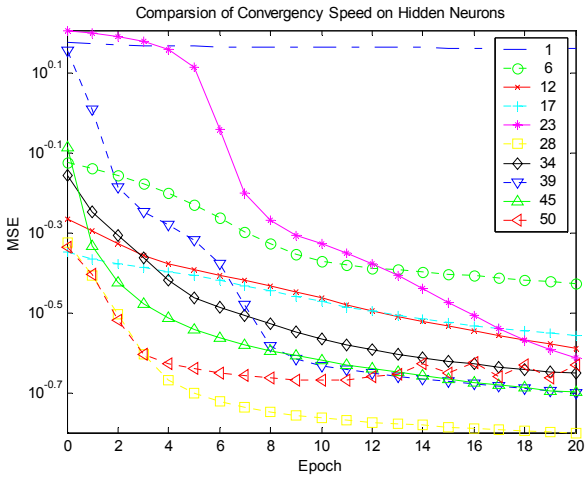

Figure 4. The comparison of a convergence speed on the size of the number of hidden neurons, the y-axis is a logarithmic scale

Figure 4 shows that about 28 hidden neurons are a good choice for constructing the hidden layer. Hence, we the following experiments would fix a number of 26 hidden neurons on all remaining training processes to construct their hidden layer. The results reveal the phenomenon that is too many hidden neurons causing this model to have more sensitivity, but small hidden neurons will result in divergence of performance. Both of them are not appropriate. The following two experiments will observe that the convergent speed and the accuracy rates of classification of subclasses with ten varied algorithms.

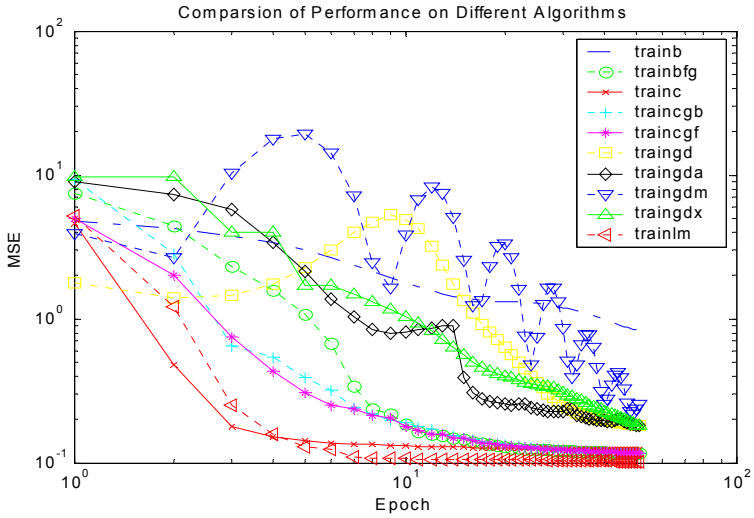

Figure 5. The comparison of a convergence speed on the varied algorithms, both x-axis and y-axis are a logarithmic scale

Figure 5 shows that Levenberg-Marquardt algorithm has the best convergent speed and the gradient steepest descent with momentum-only has a wavelet-style convergence. Figure 6 shows that the accuracy rates of classification of subclasses have the nearly same results. The results are our primary problems we would like to overcome since their skew distributions of accuracy rates whether one of those algorithms applied. The y-axis represents the accuracy rates of classification at each epoch.

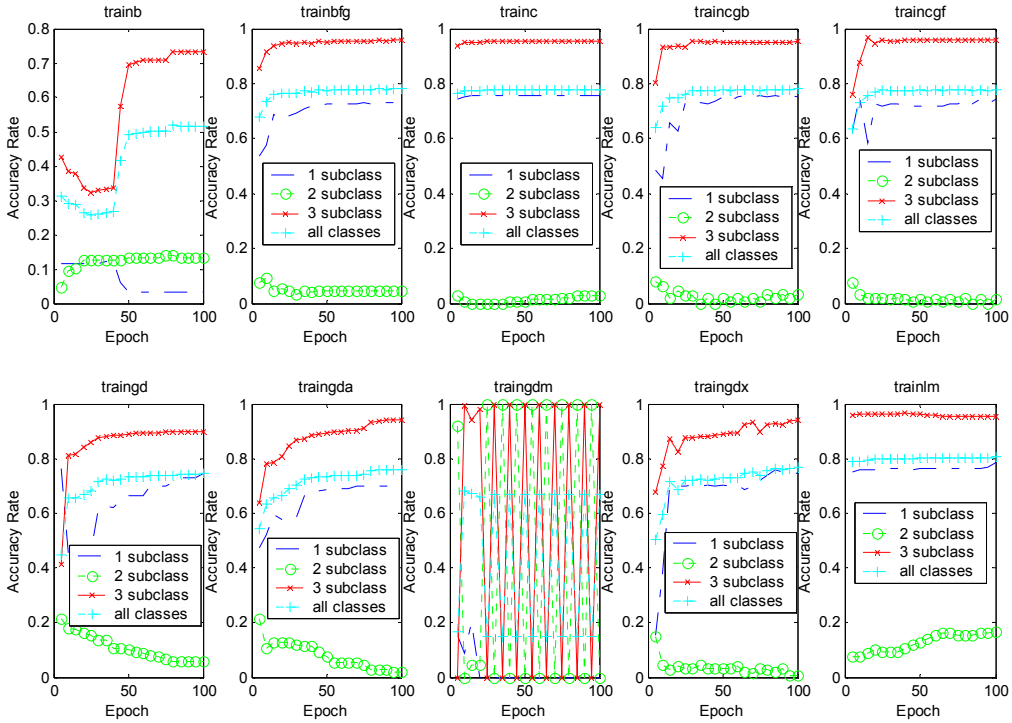

Figure 6. The comparison of the accuracy rates of classification on the varied algorithms

4.2. Noisy interference

In this experiment, we would like to have the knowledge of the model undergoing noisy signals. We therefore define two kinds of noisy data, one is randomization and the other is transformation. The randomization means that we intrude random noisy data upon the training dataset misled the model into a mistake of leaning at certain ratio of the original dataset. The transformation means that we can design a formula to transform the targeted subclasses into another ones with applicable rules. In this real case, we instruct a simulation of more severe policy of credit scoring that shifts the good customers to risk ones, shifts the risk customers to declined ones and

persist the declined customers at certain ratio of the original dataset. Here, we avoid to involve an inherently noisy property of target dataset preliminarily processed without duplication.

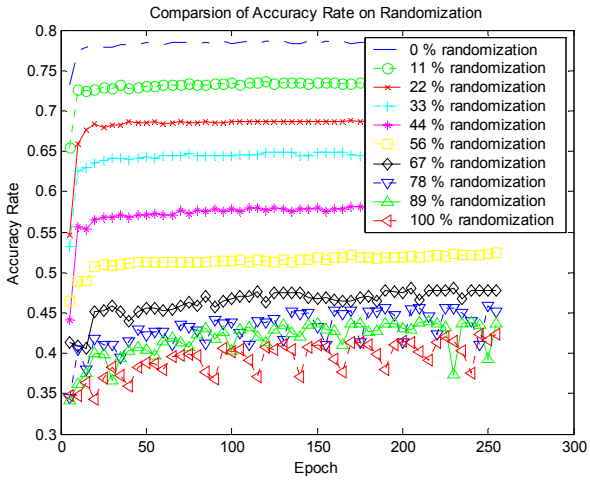

Figure 7. The comparison of the accuracy rates of classification on randomization

Figure 7 shows that the average accuracy rate decreases proportionally on the randomization. It reveals that the accuracy rate decrease 5 % by adding 10% random noised data. We then conclude that the model can undergo about average 15% noised signal so that the model still uphold the average accuracy rate of classification over 70 % roughly since the original accuracy rate is 79% with no randomization.

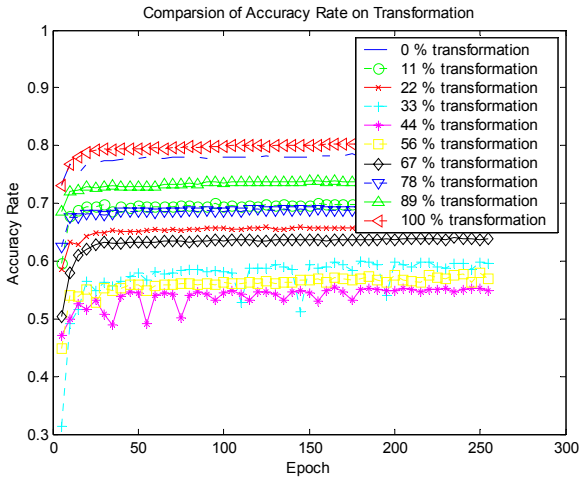

Figure 8. The comparison of the accuracy rates of classification on transformation

Figure 8 shows the 100% transformation via our mentioned rules make a little higher accuracy rate of classification than without any transformation via the same rules. Accordingly, the a pairs of (89%, 11%) and

(78%, 22%) have the same situation but make lower accuracy rate of classification averagely. A range of the middle transformation from 33% to 56%, i.e. 50 % averagely, produces the 50% accuracy rate of classification like a throwing-coin's probability. We therefore conclude that the severe policy of credit scoring is preferred the classification of two classes and the rate of distribution of good customers is unproportionate.

4.3. Data clean and training proportionality

In the following experiment, we would like to acquire the knowledge of what factors dominate the accuracy rate of classification of subclasses in the model and of why the subclass of declined customers has so tiny accuracy rate against others. Hence, we need further to clean the training datasets once again processed without any contradiction.

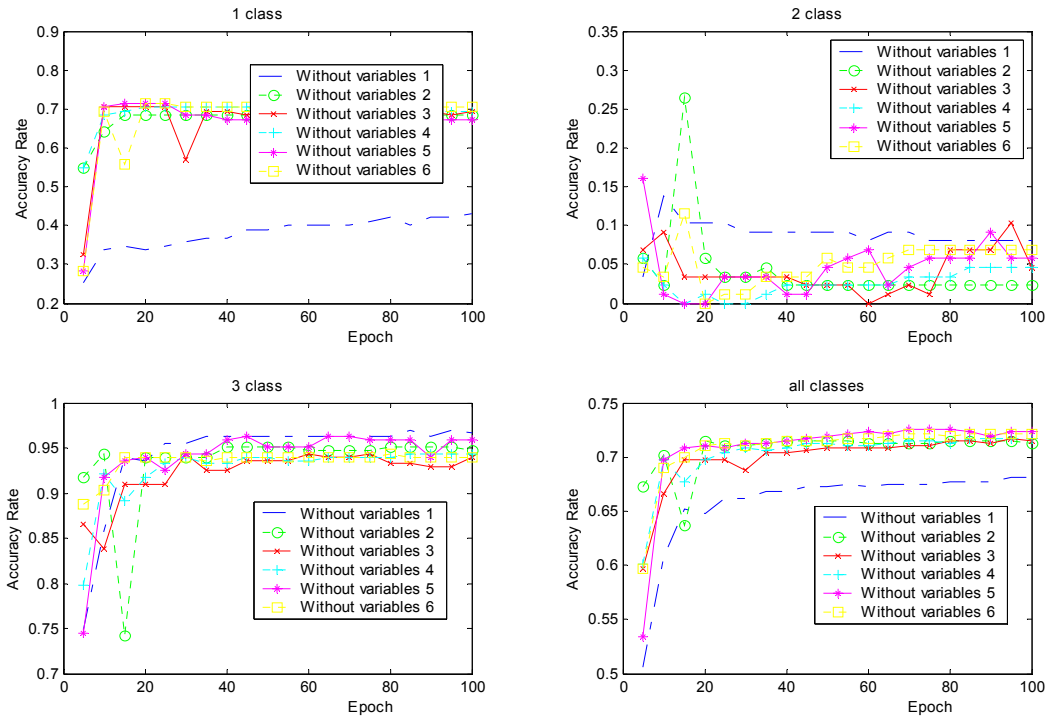

Figure 9. The comparison of the accuracy rates of classification on removing one of six variables

Figure 9 shows that the accuracy rates of classification apparently decrease when the classification of '1 class', namely declined customers, without the information of 'variable 1',namly overdraft type. That is to say that the variable of an overdraft-type dominates the accuracy rate of classification of a declined customer. This

result is reasonable since old habits hard. Due to the average accuracy rate somewhat decrease to about 72% since lacking any one of the six variables. We therefore conclude that any one of the six variables also affects the average accuracy rate of classification about 7%. However, the dominated variable '1' reaches highly bias over 22%.

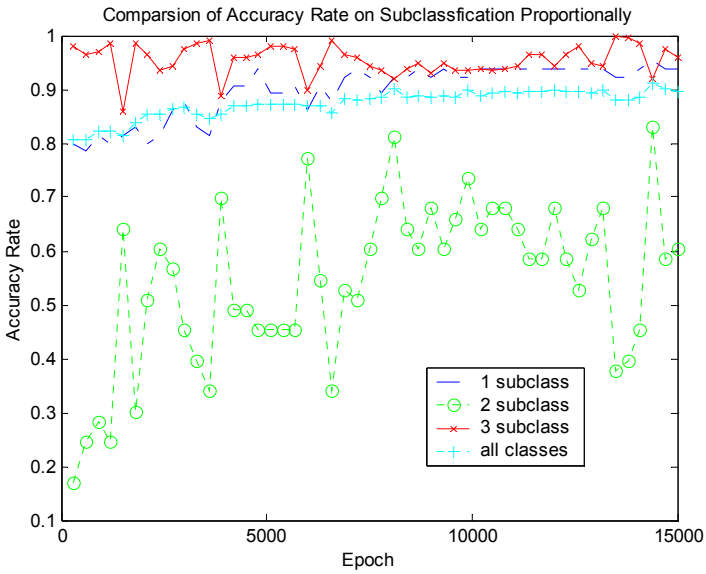

Figure 10. The results of the accuracy rates of classification on after data cleaning but without proportionality

Up to now, we have cleaned our target dataset many times, although, Figure 10 shows that the average accuracy rate has somewhat increased to around 85%. It's not a fairly increasing of accuracy rate for each subclass. Especially, the '2' class, namely risk customers', makes an extremely unstable accuracy rate of classification and ever makes ones down to 50%. It does not conform to our goals. Consequently, we select the segments of the training dataset of a proportionality ratio 1:1:1 for each subclass and then mix with each other. Figure 11 shows that the goals that we wanted are reached. That is a fairly increasing of accuracy rates on each classes raise up to over 65%. Finally, Figure 12 shows that the execute time of three algorithms which is a Levenberg-Marquardt algorithm, a quasi-Newton algorithm and an algorithm of 'Trainc' is directly related with the number of hidden neurons. Particularly, a Levenberg-Marquardt algorithm is really not suitable to construct a large networks although it can gain a highly accuracy.

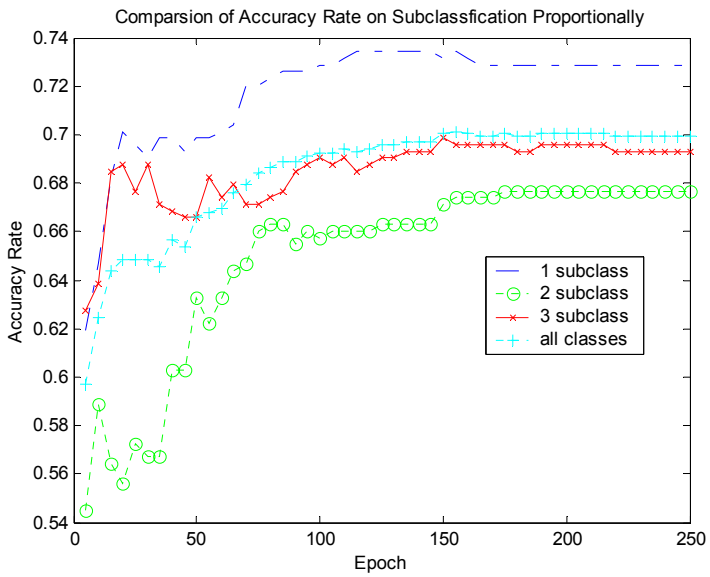

Figure 11. The results of the accuracy rates of classification of subclasses on after proportionality and data cleaning

5. Conclusions

By this study, we increase understanding by showing which factors mostly affect specific results. For the TNB, the primarily models reveal that the important factors to assess the credit scoring of customers only are with a checking account. Of course, many factors may affect a customer's credit such as loan and other individual information of transactions. Here, this training tested is merely based on the models of a small data sample. The TNB's preliminary data mining study sought to demonstrate the technology's potential as a predictor and learning tool. In the future, TNB plans to expand the scope of research for data mining including other tools expect for neural networks. Especially, it plans to include payment histories in the loan, and we hope that this data will help further improve the model's accuracy. Eventually, the TNB will use these models to identify all credits belonging to a customer with the goal of reducing late payments and defaults.

Due to the content limitation, we have not listed all experimental results yet. In the future, we will design an adaptive model to make a leaning among these parameters of the model and to increase the capacity of outside testing.

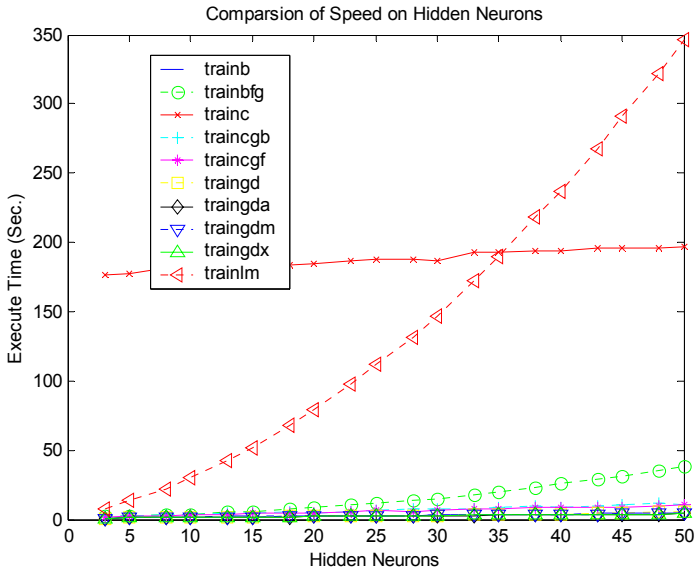

Figure 12. The comparison of the execute time on hidden neurons with different algorithms

References

[1] Gerritsen, R., “Assessing Loan Risks: A data mining Case Study,” IEEE IT Professional, Vol. 1 Issue: 6, Nov.-Dec., 1999, pp: 16-21.

[2] Lloyd-Williams, M., “Case Studies in the Data Mining Approach to Health Information Analysis,” IEEE Knowledge Discovery and Data Mining (1998/434), IEE Colloquium on, 1998, pp. 1/1-1/4.

[3] Haykin, S., Neural Networks: A Comprehensive Foundation, 2nd ed., Prentice-Hall, Ontario, Canada, 1999.

[4] Lan H., Data Mining practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, San Francisco, California, 1999.

[5] Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P., "The KDD Process for Extracting Useful Knowledge from Volumes of Data," Communications of the ACM, 39(11), 1996, pp. 27-34.

[6] Hedberg, S.R., "The Data Gold Rush" Byte, 20(10), 1995, pp. 83-88.

[7] Bates, R.R.; Mingui Sun; Scheuer, M.L.; Sclabassi, R.J., “Detection of seizure foci by recurrent neural networks,” Engineering in Medicine and Biology Society, Proceedings of the 22nd Annual International Conference of the IEEE, Vol. 2, 2000, pp. 1377-1379.

[8] Rumelhart, D.E., G.E. Hinton, and R.J. Williams, "Learning representations of back-propagation errors," Nature (London) Vol. 323, 1986, pp.533-536.

[9] Fletcher, R., Practical Methods of Optimization, 2nd edition, Wiley, New Your, 1987.

[10] Powell, M.J.D., "Restart procedures for the conjugate gradient method," Mathematical Programming, Vol.12, 1977, pp. 241-254.

[11] Pytlak, R., "A globally convergent conjugate gradient algorithm," Decision and Control, Proceedings of the 32nd IEEE Conference on, Vol.3, 1993, pp. 2890-2895.

[12] Jiang Minghu; Zhu Xiaoyan; Yuan Baozong; Tang Xiaofang; Lin Biqin; Ruan Qiuqi; Jiang Mingyan, "A fast hybrid algorithm of global optimization for feedforward neural networks," Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on, Vol. 3, 2000, pp. 1609-1612.

[13] Stan, O.; Kamen, E.W., "New block recursive MLP training algorithms using the Levenberg-Marquardt algorithm," Neural Networks, 1999. IJCNN '99. International Joint Conference on, Vol. 3, 1999, pp. 1672 -1677.