Available online at www.sciencedirect.com
SCIENCE DIRECT®

Nonlinear Analysis: Real World Applications 7 (2006) 720–747
www.elsevier.com/locate/na

Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem

Yueh-Min Huang[a], Chun-Min Hung[a,*], Hewijin Christine Jiau[b]

[a]*Department of Engineering Science, National Cheng Kung University, No.1, Ta-Hsueh Road, Tainan 701, Taiwan, ROC*
[b]*Department of Electrical Engineering, National Cheng Kung University, No.1, Ta-Hsueh Road, Tainan 701, Taiwan, ROC*

Received 28 October 2004; accepted 25 April 2005

Abstract

Most of the real-world data that are analyzed using nonlinear classification techniques are imbalanced in terms of the proportion of examples available for each class. This problem of imbalanced class distributions can lead the algorithms to learn overly complex models that overfit the data and have little relevance. Our study analyzes different classification algorithms that were employed to predict the creditworthiness of a bank's customers based on checking account information. A series of experiments were conducted to test the different techniques. The objective is to determine a range of credit scores that could be implemented by a manager for risk management. As a result, by realizing the concept of classification with equal quantities, the implicit knowledge can be discovered successfully. Subsequently, a strategy of data cleaning for handling such a real case with imbalanced distribution data is then proposed.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Class imbalance; Back-propagation; Data mining; Credit scoring

* Corresponding author.
E-mail addresses: huang@mail.ncku.edu.tw (Y.-M. Huang), goodmans@giga.net.tw (C.-M. Hung), jiauhjc@mail.ncku.edu.tw (H.C. Jiau).

1468-1218/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.nonrwa.2005.04.006

1. Introduction

Nowadays, digital information is relatively easy to find and fairly inexpensive to store, and further, several data mining approaches have been implemented to analyze it. These methods mainly include clustering, classification, association rules, and prediction. Many algorithms are used to implement the methods such as the well-known neural network approach (NN) [11], genetic algorithms, data mining tools, and fuzzy logic. However, the amount of real-world data so far collected are far less than that generated. Especially, collecting sufficient data under authorization constraints imposed by government's law proclaimed for the protection of individuals with regard to the processing of personal data is difficult. Fortunately, a particular encoded dataset encrypting some sensitive attributes was collected in a cooperative project involving the local Tainan Business Bank (TNB). A real case study can be found in [10].

First of all, our study presents many back-propagation algorithms for discovering knowledge of interest useful for TNB, and especially for finding out the optimal network parameters. Because the collected raw data have the class imbalance problem, it must be tackled properly by the known approaches [4–6,17,18,28] before using neural networks. The class imbalance problem, in which one class is represented by a large number of examples while the other is represented by only a few, is an important issue since it causes a significant bottleneck in the performance attainable by standard learning methods which assume a balanced class distribution. In 2001, Japkowicz and Stephen [14] debate in clarity several basic re-sampling and cost-modifying methods tackling the class imbalance problem, and establishing a relationship between concept complexity, size of the training set and class imbalance level. Most notably, their studies conclude that Multi-Layer Perceptrons (MLP) seem less affected by the class imbalance problem than the decision tree. Hence, our study is concerned with the comparison between MLPs and decision tree methods. However, our intent is to develop a data mining procedure remedying the class imbalance problem so that the patterns extracted from a relatively small class have a relative performance in the other classes. By conducting many experiments, only the balanced data cleaning with nearly equal subclass size clearly affects the performance at classification. In addition, the noise data, another important factor, are also considered in our study for deciding the tolerant extent of performance that could be implemented by a manager for risk management. Next, several typical data mining tools other than the neural networks are employed by applying some data cleaning procedures to understand the relationship between the noise data and class imbalance problems. Finally, this paper presents some patterns extracted from a relatively small class using the learning model, and explains its application on a credit assessment task.

In fact, the accuracy of a classification of the special or ambiguous classes is more important than other distinct classes for some business rules. In particular, real-world data often include some disproportionate subsets within a training dataset while a supervised learning rule is used to perform the overall classification. The situation will lead to supervised learning generating a model of high complexity but low interest for these special or ambiguous classes and then form a biased learning through that model. Credit scoring plays a more important role in a slacker economy. Therefore, assessing a customer's credit in loan application is a critical work for a bank. Moreover, the credit in a checking account is a primary credit check. Consequently, in this work, the datasets would be categorized into three classes—to be declined, risky, and good whose number of element of the list are class ‘1’, class ‘2’ and class ‘3’, respectively. In particular, the risky class is interesting to analyze because recognizing patterns is difficult if the classification work is determined by humans. Finally, the attributes that dominate the accuracy of the classification and the factors that dominate the accuracy of the classification of interesting classes, especially risky classes, must be known in advance.

In a classification or prediction problem, neural network techniques are used as a tool to analyze datasets. A multiplayer feed-forward network is an important class of neural networks [10]. This paper employs over ten kinds of algorithms to make an adaptive learning model and thereby predict the credit scores of checking accounts at a local bank in Taiwan. Experimentally, about sixty thousands records were collected as inputs. For efficiency, only some of the original data were retrieved to cross-analyze the complex combination of environmental parameters that affect the neural network. For each epoch, the algorithm of the neural networks would iterate the computation with same training samples. In fact, it can be found that use of some sufficient samples can simulate the behavior of the whole original data. In this study, the following requirements must be met to approach the goal. (a) The parameters of the neural network used for training in the real case must be known. They include learning rate, number of epochs, condition of convergence such as the mean square error, the properties of algorithms, the number of neurons and the time for execution. (b) The robustness of the constructed model must be tested. It includes outside test, overfitting problem [27], and tolerable noise. (c) For special purposes, the accuracy of classification for the risky class must be improved to reach an acceptable standard but not excessively affecting the accuracy of classification for other classes. The experimental processes must include basic steps of the knowledge discovery process, such as cleaning data, selecting the PCA (Principal Components Analysis) [11], pre-processing, mining data interpretation [7], and determining dominant attributes by removing a variable without affecting the accuracy of classification. For completeness, many methods of data mining are also used to compare the experimental results to verify the specific conclusions.

2. Data mining

In this section, the application of data mining used in this work is described. In general, the process of knowledge discovery, as shown in Fig. 1, is composed of four major parts of data mining. For credit assessment tasks, each part of this process will be illustrated by conducting a serial of experimental steps using the real-world bank datasets. Subsequently, the prior work depicted by a training exercise is described after the definitions of some terminologies.

2.1. Knowledge discovery process

The process of knowledge discovery via data mining can be generally divided into four activities—selection, pre-processing, mining data, and interpretation [7].

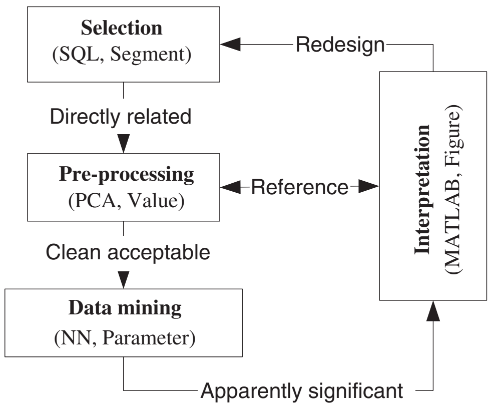

Fig. 1. Process of knowledge discovery.

Fig. 1 depicts the knowledge discovery process in this real case. The first item in parentheses shown within the block diagram of Fig. 1 is an abbreviation of the techniques used in each stage and the second item in parentheses shown within the block diagram of Fig. 1 is a major aim of the task performed in that stage. First, a Structure Query Language (SQL) tool is used to select some source data from the bank's database server. These source data, called target datasets, are cleaned in a series of pre-processes, including manual and automatic processes and a technique of PCA [11]. Moreover, MATLAB (MATrix LABoratory) Neural Network Toolbox 4.0 from MathWorks® Corp. is applied to perform the training exercises.

For selection, a target dataset of about 1000 records, which distributes the '1', '2', and '3' classes to the ratio 19:15:66, is sampled randomly from part of the original dataset of 62,621 records. The number of target records is reasonable, since nearly 1000 checking accounts are opened annually in this bank. Annually, the bank has applied several data mining tools for generating some business rules employed by the bank for the change of credit policy. Because the change influences the customer behaviors only in succeeding years, the assessment task in that year should use the dataset collected in previous years close to that year rather than using all datasets accumulated so far. During this phase, the work focuses on finding out the pattern of risky classes and cleans any inconsistencies from these selected records. During pre-processing, the configuration of datasets on the 20-field database is examined. However, some fields including irrelevant fields, private fields and missing data, are discarded. As a result, there are only six fields left: 'type of overdraft', 'type of account', 'number of transactions', 'status of recent transactions', 'interest over this year', and 'any changes to signature of the specimen seal'. These are marked as variables 1 to 6 in some figures, respectively. For avoiding unnecessary computation, this study chooses only some essential attributes using the principal component analysis (PCA) method during the data mining selection phase shown in Fig. 1. Notably, the retained fields may include skewed fields and outlying data points. Finally, data are converted into a format acceptable for a neural network. The experimental section describes further cleaning. For data mining, neural network data mining software is chosen. In particular, the types of multiplayer

Table 1
List of nominal values in nominal attributes

Variable	Attribute	Possible values of nominal attributes	Values of neurons
1	Overdraft	None, vouching	{0, 1}, {1, 0}
2	Type	Chief, sponsor, contribution, others	{0, 0, 0, 1}, {0, 0, 1, 0}, {0, 1, 0, 0}, {1, 0, 0, 0}
3	Transactions	Very high, high, middle, low, very low	>0.8, >0.6, >0.4, 0.2, $\leqslant 0.2$
4	Interest	Huge, very high, high, middle, low, very low, none	>0.5, >0.4, >0.3, >0.2, >0.1, >0.05, $\leqslant 0.05$
5	Activity	No, yes	{0}, {1}
6	Changed-signature	No, yes	{0}, {1}
	Credit	Declined(class 1), Risky(class 2), Good(class 3)	{1, 0, 0}, {0, 1, 0}, {0, 0, 1}

feed-forward networks [11] that contain several algorithms are experimentally analyzed. In the experiments, the R-coefficient [2] is used to measure how the output variable is to the target variable. Its value is close to unity, implying that the output closely fits the target. The results are then analyzed and interpreted. This phase depends on the human domain experience, a work of 'reference' that is presented by the double arrowed line of Fig. 1 performed via a person. Next, the work of the selection phase is redesigned for a data mining cycle that can refine a final result according to requirements. The experimental section describes the results of this stage.

2.2. Training exercise

A research project of TNB involves training a data mining system. As part of that training exercise, a preliminary study was performed, involving the extraction of data from the TNB data warehouse. The sampling data contain around 1000 records, which only occupy a small amount of current checking accounts, roughly 2% of the TNB's current database. Before training, input variables and output variables must be numerically encoded to enable the predictive modeling techniques that involve neural networks to be used. A learning model is constructed from this information and a pattern of the risky class of all checking accounts is extracted. This descriptive model was then suggested to the bank for risk management. The denominated variables and their detailed descriptions for this training exercise can be found in [13]. For clarity, Table 1 lists all of the nominal values used for transforming numeric attributes into nominal attributes. Notably, all of the numeric values in the original training dataset have been normalized to the interval between zero and one, inclusively.

In this work, the skewed data imply that the group of good customers is larger than the other groups. Besides the parameters mentioned above, the experimental process tests several combinations of parameters including the number of neurons in the hidden layer, the learning rate, the algorithms with the different principles, execution times, dominant attributes, noise ratios, the proportion of the number of records for each class over all training records, the effect of the cleaning of data and others. Consequently, the aim of this exercise is to establish a predictive model with three classes, whether it is feasible or not.

There are two kinds of testing that are useful for the training model. One method uses part of the training dataset as the testing dataset, which refers to the inside test. The other method uses the dataset as its testing dataset that has not submitted to the training model. For each class, the accuracy of prediction measured by a percentage is defined as the number of successful predictions over the number of submitted records. Similarly, the whole accuracy of prediction measured by a percentage is defined as the number of successful predictions over all the number of submitted records. Hence, the accuracy of prediction with outside test is an important measurement for testing the robustness of training model. In this work, if all percentages of accuracy of prediction exceed 50% for every class it is called a feasible learning model. In other words, the phenomenon of a skewed training is eliminated if the learning model is feasible. Although the ‘Risky’ class is a minority class, it brings more profits to the bank than those of the other two classes. In fact, a high accurate prediction for the model class ‘Good’ is meaningless in terms of the imbalanced class distribution since the prediction is probably only a random guess. Therefore, a tradeoff between profits and accurate predictions should be reconsidered in the real world. Hence, the study aims at successful skewed training via a feasible learning model for analyzing the nonlinear classification problem in the real world.

3. Comparison of tools

This section introduces some typical methods for knowledge discovery and data mining. These methods, which differ from the methodologies of neural networks, can extract interesting information or patterns from a large database. For classification learning, the training dataset used for this real case must slightly modify the designed schema to make them suitable for various classification methods. First, the schema of the training dataset is composed of the seven attributes of variables, including overdraft, type, transactions, interest, activity, changed-signature, and credit. This study hypothesizes that the training dataset comprises a unique class attribute and numerous column attributes, and gathers many instances to describe the universe of credit risk related to this real case. Each instance of the training dataset is associated with a unique and predefined class value. The values in the class attribute include ‘Decline’, ‘Risky’, and ‘Good’, and are used to indicate different levels of risk, respectively. Furthermore, the attributes of a numeric type must first be discretized since some classification algorithms only deal with nominal attributes. Thus, some nominal values are applied to obtain a small number of distinct ranges. By observing data distribution, some business rules can determine reasonable ranges of discretizing values. For example, a ‘huge’ contribution of interest for checking accounts usually comes from some special customers who deposit considerable money on the very early date and then draw on the checks on the later date. Thereby, the bank can employ the money without the payment of interests since a checking account is a free-interests account. In fact, this group of special customers is well known to bank employees, and a simple summary can easily determine the number of instances. Assume that the training dataset, which is an information table, is used to describe the universe of knowledge for this real case. The information table consisting of many instances can be partitioned by dividing the instances into disjoint classes. Assume that the set of attributes $\mathscr{A} = \mathbb{F} \cup \mathbb{C}$ constructs the information table S, where the set $\mathbb{F}$ of attributes is used to describe the instance $s_j \in S$ and $\mathbb{C}$ is a class attribute. The goal is to find classification rules of the form, $\chi \Rightarrow \mathbb{C} = c_i$ where χ denotes a formula over $\mathbb{F}$ and c_i is a class value. The formula χ can describe the set $\omega(\chi) = \{s_j \in S | \forall s_j \text{ satisfy } \chi\}$ of all instances with the property expressed by the formula χ. Ordinarily, a subset $\hat{\mathscr{A}} \subseteq \mathscr{A}$ can define either a partition π_A or covering τ_A of the universe. To be complete, this real case must use various methods based on both partition π_A and covering τ_A to verify the experimental results using back-propagation learning. Therefore, this study briefly compares the algorithms of C4.5, ID3, PRISM, Conjunctive Rule, NNge, IBk, Naı¨ve Bayes, Complement Naı¨ve Bayes, Bayes NetB, Random Committee, Voted Perceptron, and RBF Network, as well as the multiple perceptron of neural networks mentioned above.

3.1. Back-propagation learning

This section briefly introduces the well-known ten algorithms for back-propagation learning [25], applied in the experiments. The error back-propagation learning consists of two passes, one of which propagates the effect from the input layer through the hidden layers to the output layer, and then the other pass propagates an error signal backward through the network by an inverse direction of the original pass. The best simple back-propagation algorithm is the gradient steepest descent algorithm. Besides, the transformations of gradient steepest descent methods are employed to train and test the dataset, including ‘Traingd’, ‘Traingda’, ‘Traingm’ and ‘Traingx’. Moreover, the weight and bias in algorithms like ‘Trainb’ and ‘Trainc’ are modified in batch and online modes, respectively. Besides, a conjugate gradient method [19] belongs to a class of second-order optimization methods known collectively as conjugate-direction methods. Thereof, the Fletcher-Reeves formula and the Powell–Beale Restarts methods are also tested experimentally by the algorithms of ‘Traincgf’ and ‘Traincgb’, respectively. Additionally, a quasi-Newton method only estimates the gradient vector and uses second-order information on the average error energy function to update the approximated Hessian matrix $\mathbf{H}'$ without any knowledge of the Hessian matrix $\mathbf{H}$. The method is assured of going downhill on the error surface [8]. A quasi-version of the Newton method is therefore used and denoted by ‘Trainbfg’ in the experiments. Finally, the Levenberg–Marquardt algorithm [26] is designed in a class of second-order optimization methods and is similar to a quasi-Newton method, but it lacks the computation of a Hessian matrix. ‘Trainlm’ denotes the Levenberg–Marquardt algorithm in the experiments.

For a given training dataset, the average error energy $\Re_{\text{av}}$ is defined as a function of the weights and bias denoted by vector $\mathbf{w}$. The vector $\mathbf{w}$ is a set of parameters that describe neurons of this network. The aim of the learning process is to minimize $\Re_{\text{av}}$. Hence,

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

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

The gradient steepest descent method computes the gradient from the first derivative of $\Re_{\mathrm{av}}$ with respect to w. Therefore,

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

The best simple back-propagation algorithm is the gradient steepest descent algorithm. The chain rule of Calculus is used to provide a mathematic deduction for modifying weight and bias,

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

The transformations of gradient steepest descent methods are employed to train and test the dataset, including ‘Traingd’, ‘Traingda’, ‘Traingm’ and ‘Traingx’. Moreover, the weight and bias in algorithms like ‘Trainb’ and ‘Trainc’ are modified in two modes; one is the increment mode that changes the vector **w** of weight and bias when one column vectors of input variables are submitted, while the other is the batch mode that changes the vector **w** of weight and bias when the whole input vectors are submitted.

3.1.1. Conjugate gradient

The mentioned methods adjust the weight vector **w** in the direction of deepest descent. These methods may not quickly reach convergence for the error function. The conjugate gradient method [19] belongs to a class of second-order optimization methods known collectively as conjugate-direction methods. Fast convergence is necessary for back-propagation learning. This method can accelerate the typically slow convergence experienced with the method of steepest descent and must compute a Hessian matrix $\mathbf{H}(n)$ defined as follows.

$$\begin{aligned}\mathbf{H}(N) &= \frac{\partial^2 \Re_{\mathrm{av}}}{\partial \mathbf{w}^2}\\ &= \frac{1}{N}\sum_{n=1}^{N}\left\{\left(\frac{\partial F(\mathbf{w}, \mathbf{p}(n))}{\partial \mathbf{w}}\right)\left(\frac{\partial F(\mathbf{w}, \mathbf{p}(n))}{\partial \mathbf{w}}\right)^r - \frac{\partial^2 F(\mathbf{w}, \mathbf{p}(n))}{\partial \mathbf{w}^2}(t(n) - a(n))\right\}.\end{aligned} \tag{4}$$

The conjugate gradient algorithm can avoid using a Hessian matrix $\mathbf{H}(n)$, which is plagued by computational difficulties, to determine the next direction of the search. Therefore,

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

where $s(n)$ represent a conjugate vector in the direction of the search. Here, two major ways to evaluate the coefficient $\beta(n)$ of the conjugate gradient are chosen to determine the direction of search to minimize the average error energy function $\Re_{\mathrm{av}}$ without explicit knowledge of the Hessian matrix $\mathbf{H}(n)$. One method is to use the Fletcher–Reeves formula [21,15] as follows.

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

where $\mathbf{r}^{\mathrm{T}}$ is a transpose vector of a gradient vector $\mathbf{r}$. Another way is to use the Powell–Beale Restarts formula [19] as follows:

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

where 0.2 is an orthogonality ratio that is the orthogonal quality between the current gradient vector and the preceding gradient vector. This formula implies that the direction of search will periodically be reset to a direction of a negative gradient if a certain and tiny orthogonality exists. The Fletcher–Reeves formula and the Powell–Beale Restarts methods are tested experimentally by the algorithms of 'Traincgf' and 'Traincgb', respectively. Here, another method is introduced for solving a combinatorial optimization problem. This method is called a Newton method, whose 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.

However, the method is impractical because the computation of an inverse Hessian matrix $\mathbf{H}^{-1}$ is difficult. A quasi-version of the Newton method is therefore used and denoted by 'Trainbfg' in the experiments. The method 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 estimates the gradient vector $\mathbf{r}$ and uses second-order information on the average error energy function to update the approximated Hessian matrix $\mathbf{H}'$ without any knowledge of the Hessian matrix $\mathbf{H}$. The method is assured of going downhill on the error surface [8].

3.1.2. Levenberg–Marquardt algorithm

The Levenberg–Marquardt algorithm [26] is designed in a class of second-order optimization methods and is similar to a quasi-Newton method, but it lacks the computation of a Hessian matrix. A Hessian matrix is approximately computed by the formula $\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$ and the gradient $\mathbf{r} = \mathbf{J}^{\mathrm{T}}\mathbf{e}$ if $\Re_{\mathrm{av}}$ is formed by a summation of squares, where $\mathbf{J}$ is a Jacobian matrix which includes the first derivative of $\Re_{\mathrm{av}}$ with respect to $\mathbf{w}$ and $\mathbf{e}$ is the error vector of the network. Eq. (8) is rewritten and combined with the 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 $\mathbf{I}$ is an identity matrix. The Levenberg–Marquardt method switches its equivoque between the quasi-Newton method and the smaller gradient descent method, by adaptively tuning the μ-value from zero to very high or from very high to zero, while $\Re_{\mathrm{av}}$ changes from decreasing to increasing or from increasing to decreasing. The Levenberg–Marquardt algorithm is more efficient in converging a minimum error than others. However, the computation of a Jacobian matrix $\mathbf{J}$ in the algorithm is more complicated than the computation of a Hessian matrix $\mathbf{H}$ in the quasi-Newton method. Also, the implementation of the algorithm requires considerable memory for storing the Jacobian matrix $\mathbf{J}$ of size $Q \times n$, where Q is the number of records in the training dataset and n is the dimensionality of weight vector $\mathbf{w}$. Fortunately, the Jacobian matrix $\mathbf{J}$ can be decomposed into the following form:

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

The sub-items are summed in this decomposition of the matrix for reducing the memory requirement. The experiments performed here did not test for tuning the μ-value of the Levenberg–Marquardt method since the memory consumed is not so critical at present.

3.2. RBF network

To achieve a complete comparison, the radial-basis function (RBF) network, which was surveyed early in 1985 [20], can provide a completely different approach by viewing the neural network design. The RBF network uses interpolation of the test data in a high-dimensional space to solve the curve-fitting problem. Generally, the construction of the RBF network involves three layers with completely different roles. The input layer nodes connect to the nodes of the only hidden layer in the network via a nonlinear transformation from the input space to the hidden space. The output layer then supplies the network response to the signal applied to the input layer. The hidden layer comprises N Green's functions $\mathbf{G}$ which provide the distinct data points $x_1, x_2, \ldots, x_N$. The multivariate Green function $G(x, x_i)$ is defined as follows:

$$G(x, x_i) = \exp\left(-\frac{1}{2\sigma_i}\|x - x_i\|^2\right), \tag{11}$$

where x_j denotes the center of the function and σ_i represents its width. The one-to-one correspondence between the training instance x_i and the Green function $G(x, x_i)$ creates the need for expensive computation costs. The K-means clustering algorithm may thus provide fewer basis functions for learning the considerable instances of the training dataset.

3.3. C4.5 and ID3

The C4.5 proposed by Quinlan [22] is a well-known algorithm based on an approach, which divides and conquers a problem to construct a decision tree recursively. First, the approach selects and places an attribute at the root node to generate one branch for each possible value of the attribute. The criterion for attribute selection involves obtaining a maximum information gain using the information theorem [27]. Next, the branches can split the instances into numerous partitions, including one for every attribute value. Finally, each partition recursively repeats the splitting process until all instances at a node share the same classification. To simplify decision tree size, a pruning strategy is also applied to cut the nodes with rarely statistical significances throughout all the experiments using the C4.5 algorithm. Similarly, the other partition approach ID3 [27] can construct a similar decision tree using the instance-based method without pruned nodes. The approach selects an attribute to split the universe such that the training dataset is divided into many non-empty subsets, one for each attribute value. Subsequently, the repeating steps are recursively applied to every branch until all instances in the training set belong to the same class.

3.4. PRISM

Regarding covering approaches, the PRISM method proposed by Jadia Cendrowska in 1987 [3] is a typical covering algorithm for generating rules. Instead of using the principle of generating decision trees, which can be converted to decision rules, PRISM generates rules from training dataset directly. Due to PRISM providing shorter rules than ID3, and being suitable for non-contradiction cases the method applied to the real case would be a very interesting method of handling contradiction rules. By combining two different approaches, partition and covering, the resulting experiments may provide more information for understanding the behavior of risky customers crediting a class value of 'Risky'. Conceptually, this approach calculates the probability that class c_i will occur for each attribute-value pair $\mathscr{AV}_j$. Depending on the maximum probability of $\mathscr{AV}_j$, a subset of the training set is created that comprises all instances selected by the $\mathscr{AV}_j$. The two steps are repeated for this subset until it contains only instances of class c_i. Subsequently, the conjunction of all the selected $\mathscr{AV}$ as creating this subset yields an induced rule. The next step removes all instances covered by this rule from training dataset. The steps continuously repeat until all instances of every class c_i are removed from the training dataset.

3.5. Conjunctive rule

For further comparison of rule-based and tree-based classification, the discussion of performance should involve the conjunctive rule approach since decision trees have difficulty in expressing the disjunction implied between the different rules in a set. In fact, a disjunction form of an expression can be rewritten by transforming the logic expression into the disjunctive normal form that is a disjunction of conjunctive conditions. Thus, a conjunctive rule learner may also be introduced to predict numeric and nominal class value. The conjunctive rule approach reads a set of rules directly off a decision tree. One rule is generated for each leaf on the tree. The path from the root to that leaf includes the antecedents of the rule and this is consequent of the final leaf node assigned class value. The antecedents of the rule are conjunctive with logic AND, and the consequents are the available class values in the attribute class. If this rule does not cover a test instance, the default class serves as a predicted value. Similarly, this approach also calculates an "Information" gain and thus selects an antecedent. The "Information" of one antecedent is the weighted average of the entropies of both the data covered and that not covered by the rule for this classification problem. Furthermore, a simple pre-pruning strategy based on the number of antecedents can prune the generated rule to produce a shorter rule.

3.6. NNge and IBk

Additionally, the nearest-neighbor-like algorithm using non-nested generalized exemplars (NNge) is one of the instance-based approaches. The generalized exemplar indicates high-dimensional rectangular regions of the instance space. Most of the instance-based classifications use Euclidean distance to compare new instances with existing ones, and use the nearest existing instance to assign its class value to the new one. In a sense, learning of instance-based approaches is very easy to carry out, since the approach merely calculates the distance in a hyper-rectangle when classifying new instances. Then the approach merges correctly classified instances with the nearest exemplar of the same class, and alters the boundaries of the hyper-rectangle so that it shrinks away from the new instance if the prediction is incorrect. Besides, the non-nested version can avoid a single rectangle covering most of the instance space. Another instance-based approach is the K-nearest-neighbor classifier [1]. This approach selects appropriate K values based on cross-validation, and also conducts distance weighting. The need to scan the entire training dataset when classifying each test instance makes the classification of large training datasets extremely time-consuming. Thus, reducing the number of redundant exemplars can improve the time performance. Moreover, higher K can enable the handling of more noise. Although higher K may simultaneously raise the issue of time-consuming computation, it can achieve excellent predictive performance.

3.7. Naı¨ve Bayes, Complement Naı¨ve Bayes, and BayesNetB

Conventionally, the experimental results must be compared with a well-known Naive Bayes [16] classifier, as in [10]. The Naı¨ve Bayes method is based on the Bayes rule of conditional probability, which assumes that attributes are independent and that each attribute exerts an identical influence on the decision. In fact, the Complement Naı¨ve Bayes method can tackle the poor assumption of Naive Bayes [23] using heuristic solutions. Moreover, this comparison of prediction accuracy comprises a special model of the Naı¨ve Bayes method called BayesNetB. The BayesNetB approach is a Bayes network [12] learning algorithm that uses a hill-climbing algorithm with no restrictions on the order of variables. The Bayes network applies Bayesian analysis without assuming independence.

3.8. Random committee and voted perceptron

To provide a further comparison, the analysis reports the prediction accuracy for this real case using an ensemble of randomizable base classifiers and a voting system. In the random committee classifier, the seeds generating different random numbers build various base classifiers. The individual base classifier determines the final prediction using a straight average of the predictions for every base classifier. Additionally, the voted perceptron algorithm is based on the well-known perceptron algorithm of Rosenblatt [24]. Furthermore, Freund and Schapire [9] implemented the voted perceptron algorithm in 1998. First, the algorithm globally replaces all missing values, and transforms nominal attributes into binary ones. Second, a sophisticated method applies the online perceptron algorithm to batch learning. The algorithm retains the list of all prediction vectors during training to generate better predictions on the test data. For each such vector, the algorithm accumulates the votes of correctly classified predictions until a misclassified prediction occurs. The number of votes is then used to weight the prediction vector. Eventually, the algorithm determines a final prediction using the weighted majority votes.

4. Experimental data

A series of experiments have been conducted for verifying the correctness of the methodology described in the previous section, including the setup, interference by noise, the cleaning of data, and the adjustment of training records.

4.1. Setup and basic experiments

The network was trained and tested in the PC environment of 512 MB RAM, a 1.5G Pentium IV CPU on the Windows 2000 platform. Now, suppose a target dataset has been selected from the source database and the input vector has been formatted as a numeric matrix required by the tools. The performance of these networks is analyzed using various algorithms by measuring the value of mean square error in the experiment. In the very beginning, the first step in the experiments is to determine the range of learning rates since knowledge of these analyzed datasets may be absent.

Experiment 1. Fig. 2 shows that an initial learning rate of nearly 0.2 is preferred, includingas a benchmark for various algorithms. It also shows that the performance diverges at a rate of over about 0.4. Hence, 0.2 was fixed as the initial learning rate for the remaining experiments. The x-axis represents the number of the epoch, such that the training processes repeat 50 times throughout the entire training dataset. The y-axis represents the mean square error at each epoch.

Experiment 2. The experiment includes a method similar to that mentioned above. Knowledge of the number of hidden neurons that suffices for most algorithms is desired, because the number of hidden neurons influences the capacity for fitting non-linear curves to those

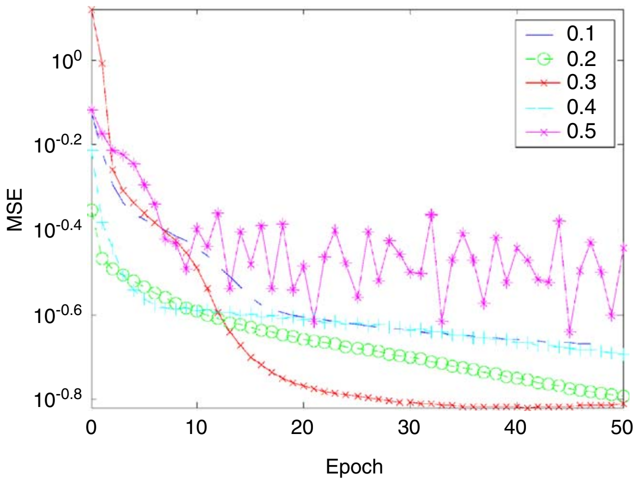

Fig. 2. Comparison of a convergence speeds at various learning rates.

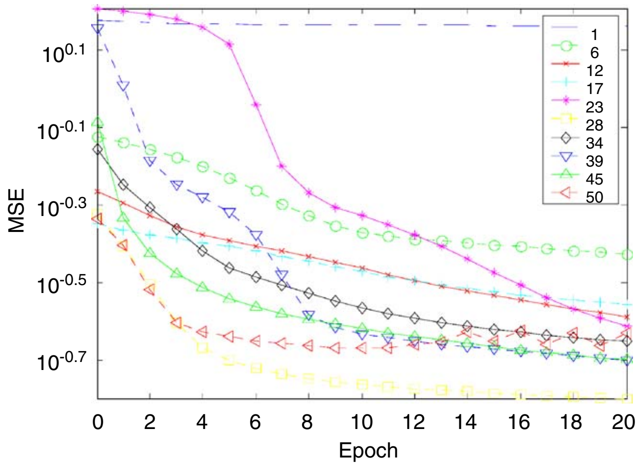

Fig. 3. Comparison of convergence speeds for various numbers of hidden neurons.

multi-dimensional variables. Fig. 3 indicates that around 28 is a good number of hidden neurons for constructing the hidden layer.

Hence, in the following experiments, a fixed 28 hidden neurons are used in all the remaining training processes to construct hidden layers. The results reveal that too many hidden neurons increase the sensitivity of the model, but very few hidden neurons cause the performance to decline or diverge. Neither situation is appropriate.

Experiments 3 and 4. The following two experiments show the speed of convergence and the accuracy of classification of classes obtained using ten algorithms. Fig. 4 shows that Levenberg–Marquardt algorithm converges the fastest and has the gradient steeply descending gradient. The gradient descent algorithms with momentum-only exhibits a wavelet-style convergence. Fig. 5 shows that regardless of any algorithm, the accuracy of classification of classes is almost the same in the circumstances set up without cleaning of data. The improvement of the accuracy of class ‘2’ is the primary work to overcome. The y-axis represents the accuracy of classification at each epoch.

4.2. Interference by noise

An experiment is conducted to yield knowledge of a training model with noisy signals. Two kinds of noisy data are thus defined—interfering with the value of goal attribute by random noisy data and changing the value of goal attribute by transformed rules. In the first instance, random noisy data are added to the training dataset to mislead the model into making a mistake with part of the original dataset. In the second instance, a formula is designed to transform the targeted classes into other ones according to a set of rules. In this real case, a more severe credit scoring policy is simulated that shifts good customers

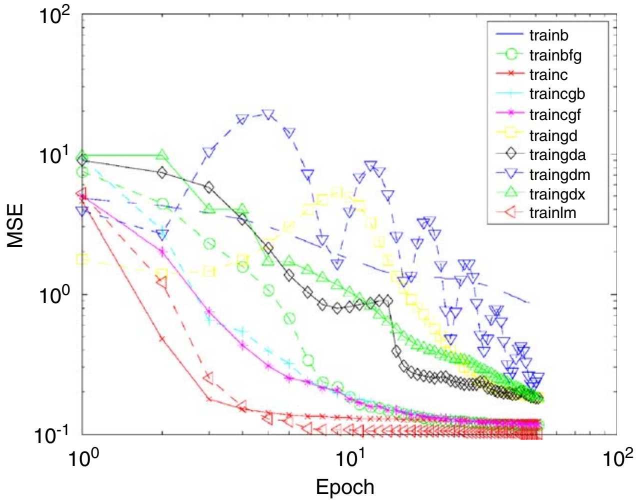

Fig. 4. Comparison of speed of convergence for various algorithms.

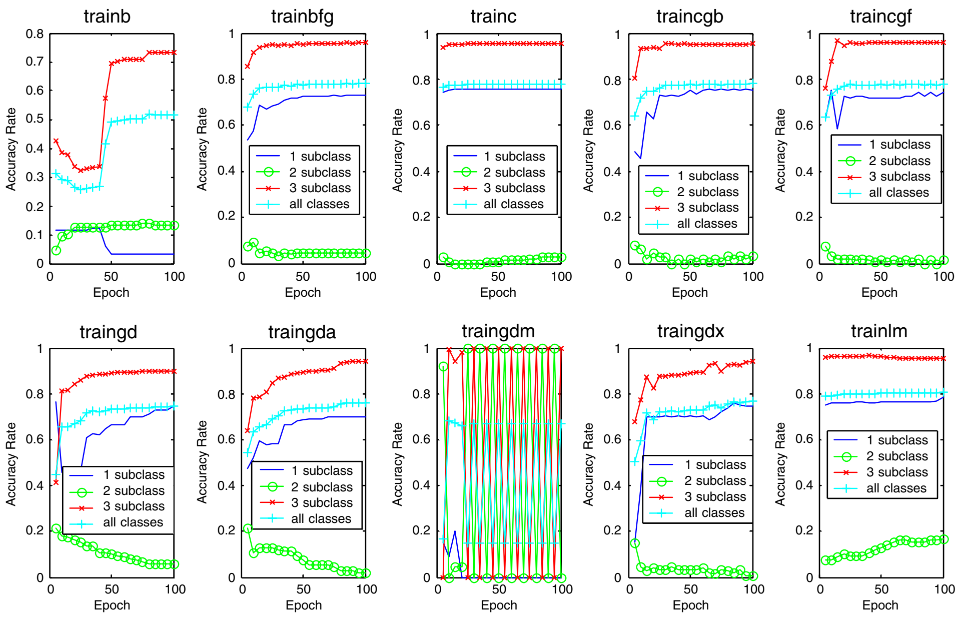

Fig. 5. Comparison of accuracy of classification for each class using various algorithms.

to risky ones and risky customers to declined ones, while a certain proportion of declined customers from the original dataset is maintained. Here, an inherently noisy target dataset needs to be preliminarily processed.

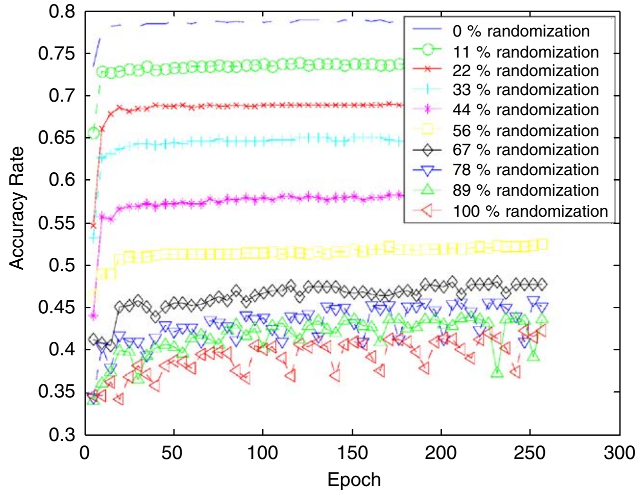

Fig. 6. Comparison of the accuracy of classification with randomization.

Table 2
Prediction performance using various methods of data mining for the classification of three classes

Methods	Accuracy %	TP rate			
		'Declined	'Risky'	'Good'	MSE
Multilayer Perceptron	81.54	0.770	0	0.973	0.1024
C4.5	81.54	0.769	0	0.974	0.1043
Naı¨ve Bayes	81.50	0.768	0.005	0.972	0.1027
Bayes NetB	81.44	0.771	0.007	0.97	0.1021
IBk	81.15	0.774	0.016	0.964	0.1041
ID3	80.99	0.774	0.017	0.964	0.1041
Conjunctive Rule	80.97	0.770	0	0.965	0.1075
Complement Naı¨ve Bayes	80.59	0.770	0.013	0.958	0.1294
RBF 200-clusters	80.42	0.699	0.005	0.974	0.1056
Random Committee	78.68	0.741	0.062	0.929	0.1176
RBF, 30-clusters	78.54	0.592	0	0.974	0.1142
RBF 3-clusters	73.84	0.259	0	0.989	0.1322
RBF 2-clusters	72.43	0.157	0	0.994	0.1359
NNge	70.26	0.629	0.124	0.825	0.1982
PRISM	21.45	0.977	0.055	0.056	0.5236

Experiment 5. Fig. 6 shows that the average accuracy decreases in proportion to the randomization.The accuracy decreases by 5% when data that are 10% random data of original dataset are added. Generally, different tools applied to this real-world dataset generate vary differences in accuracy which vary within 10%, as listed in Table 2. Because the generic neural network without using randomization data yield about 79% classification accuracy

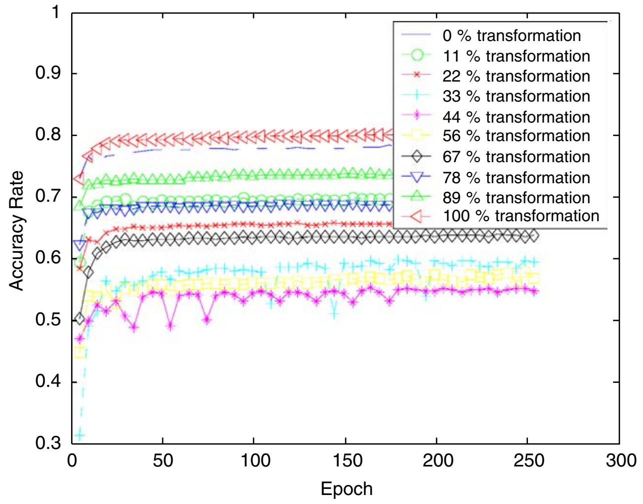

Fig. 7. The comparison of the accuracy rates of classification on transformation.

shown in Fig. 6, this study assumes that the accuracy decreased is tolerable if the training model still maintains an average classification accuracy over 70%. Correspondingly, this assumption is true only when the added noise data are less than 15%.

Experiment 6. Fig. 7 shows that 100% transformation by the stated rules slightly increasesthe accuracy of classification over that obtained without any transformation. Moreover, 89% and 11% transformation have the same accuracy of classification. It seems to connect a complementary relationship between the degrees of noise transformation. A moderate range of transformation from 33% to 56% with an average of 50% produces 50% accuracy of classification as would be obtained by throwing a coin. The results shown in Fig. 7 conclude that the severe credit scoring policy is preferred for this real case when classifying customers into only two classes and the distribution of good customers is larger than risky ones.

4.3. Data cleaning

Experiment 7. The following experiment is performed to determine which factors dominate the accuracy of classification of classes in the training model and to understand why the class of declined customers is associated with a much lower accuracy than other groups. Hence, the training datasets must be cleaned further and processed again. Fig. 8 shows that the accuracy of classification apparently decreases when the classification into class ‘1’ of declined customers is made without information on ‘variable 1’, namely type of overdraft. In each subdiagram of Fig. 8, when the dataset without variable 1 is trained, the blue dotted lines clearly present a worse prediction accuracy than that of the other cases. This result is reasonable since the class of ‘Declined’ customers inherently lacks the

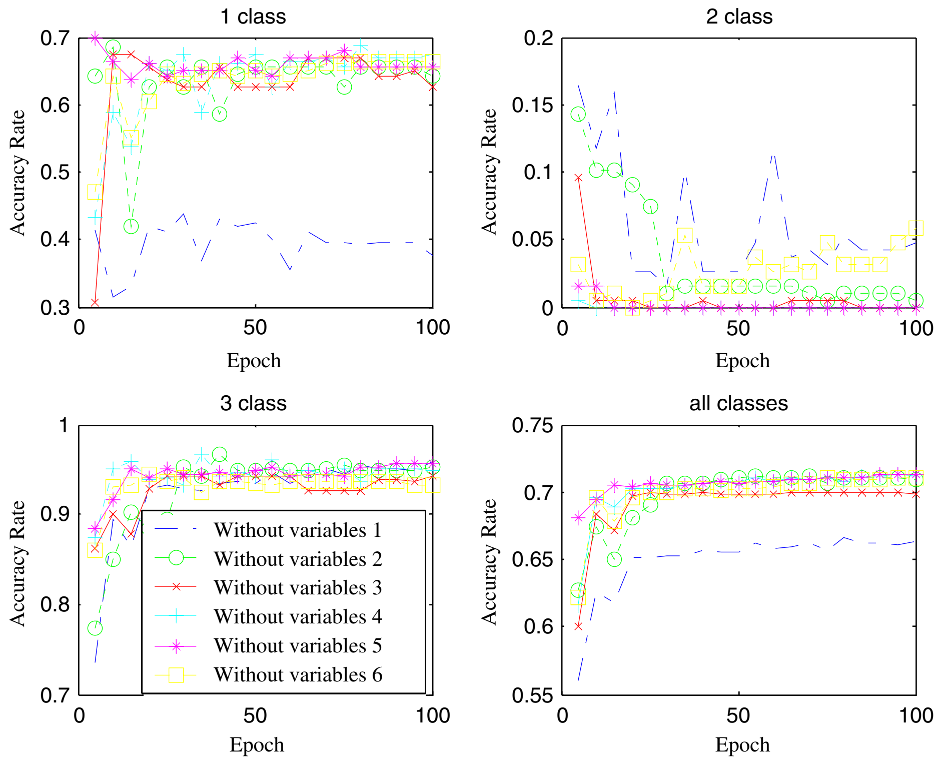

Fig. 8. Comparison of classification accuracy on removing one of six variables.

risk management intelligence exercising an overdraft right (‘variable 1’). Furthermore, the average accuracy falls to about 72% since when any one of the six variables is lacking. Therefore, any one of the six variables also affects the average accuracy of classification by around 7%. However, lacking the dominant variable ‘1’ will cause 22% bias of original accuracy.

Experiment 8. In the preceding discussion, the target dataset was cleaned many times. Fig. 9 shows that the average accuracy rate has somewhat increased to around 85% after cleaning. Especially, the accuracy of the classification of risky customers into class ‘2’ class is very unstable and can fall to 50%. This result fails to meet requirements.

Experiment 9. Consequently, the sampling segments of the training dataset are selected in a ratio 1:1:1 for each class and then mixed in random orders with each other. Fig. 10 shows that the set goals are reached. That is, the accuracy for each class is increased to over 65%.

Experiment 10. Unfortunately, requirement (b) is not met in full to approach the goal of this work since the feature of risky class is not apparently detected yet. Fig. 11 reveals that the accuracy of classification using outside test for the risky class falls to about 22%, even

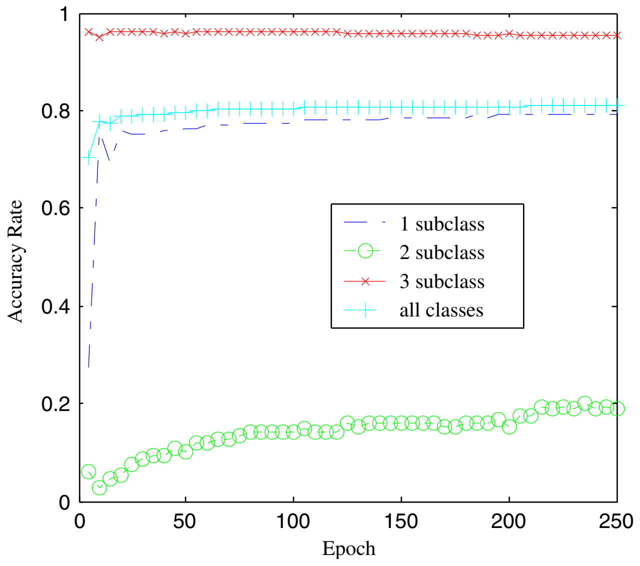

Fig. 9. Accuracy of classification after data cleaning with skewed distribution data.

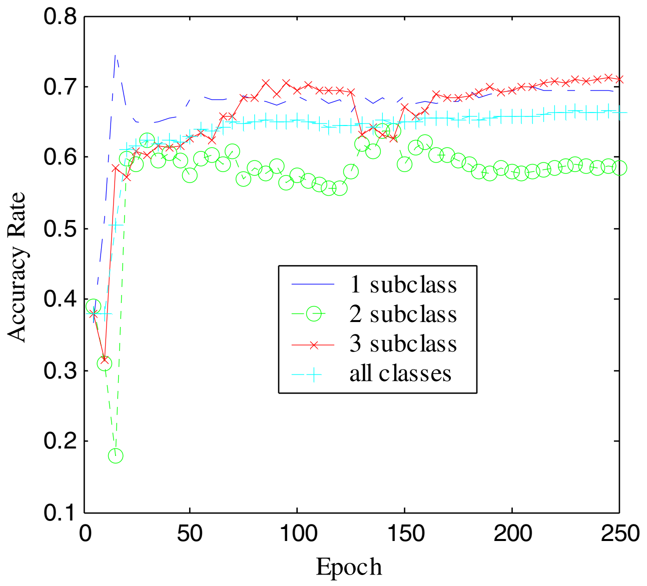

Fig. 10. Accuracy of classification after adjusting their distribution and data cleaning.

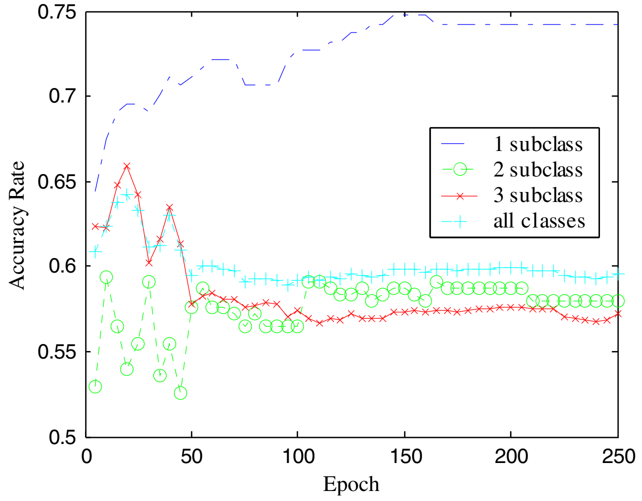

Fig. 11. Results for outside test after cleaning data and adjusting distribution of data.

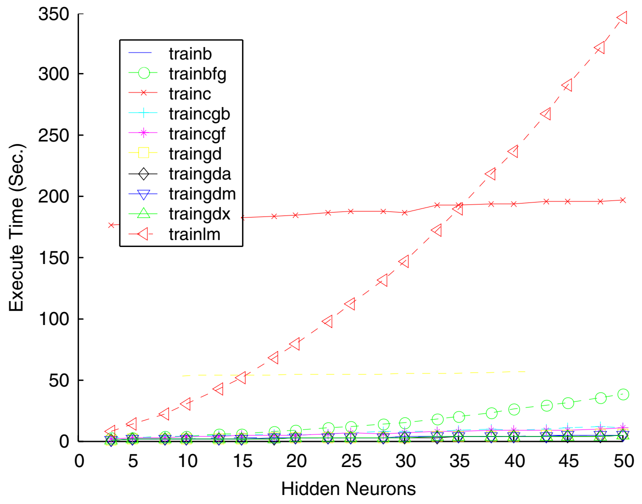

Fig. 12. Comparison of execution time of algorithms for various numbers of hidden neurons.

if the training datasets have been cleaned by the proposed model and adjusted for their distribution of data.

Experiment 11. Finally, Fig. 12 shows that the execution times of the three algorithms, a Levenberg–Marquardt algorithm, a quasi-Newton algorithm and an algorithm marked as 'Trainc' are directly related to the number of hidden neurons. In particular, a

Levenberg–Marquardt algorithm is really not suitable for constructing a large network although it can yield a high accuracy.

4.4. Comparative analysis

This section reports the accuracy, true positive rate, and mean square error (MSE) of prediction using many methods of data mining for the real case containing 10,000 instances, which distributes the '1', '2', and '3' classes to the ratio 70:13:17, rather than the 1000 previously used. The goal is to demonstrate whether the conclusions being consistent. Simultaneously, the findings of new knowledge can be further inferred using consistent results.

Experiment 12. The following experiments use the same training dataset and estimate all of parameters of performance using 10-fold cross-validation. Table 2 summarizes the prediction accuracy using various data mining techniques. From observation of the prediction accuracy, the Multilayer perceptron with back-propagation algorithm (BPN) and C4.5 have higher accuracy than the other methods. However, from the perspective of time complexity, C4.5 overwhelms BPN because of each split considered in the entropy method only taking $O(\mathrm{m} \log \mathrm{m})$ time, where m denotes the number of instances in the training dataset. Although IBk and ID3 are slightly less accurate than either C4.5 or BPN, they can identify some 'Risky' classes of interest. In fact, even if one simply uses Bayes-like methods, the classifiers can attain an acceptable accuracy as well as C4.5 and BPN. Additionally, by comparing these Bayes-like methods, the TP rate of 'Risky' gradually increases from 0.005 to 0.013. This increase indicates that the classification for the risky customers is not so dependent on attributes due to the Complement Naı¨ve Bayes method modifying the poor assumption of Naive Bayes. The analytical results indirectly demonstrate that modifications of the assumptions, all attributes are independent and consistent of affection, are necessary using Bayes-like methods for this real case. Moreover, the Conjunctive Rule can obtain the same accuracy as ID3 because it is essentiality induced from decision tree. Besides the approximation property of RBF networks, more numbers of K-clusters apparently influence prediction accuracy increasingly. Their prediction accuracy varies from 72.43% to 80.42% with the cluster number from 2 to 200. Experimentally, the Random Committee method achieves the same prediction accuracy for RBF networks with K values between 30 and 50 clusters. Before data cleaning, most neural networks, including BPN and RBF, are insensitive to the variations of the resulting TP rate. Basically, predictions using the above methods can reach an accuracy of up to around 80%. However, the prediction using the NNge method yields an accuracy of just 70.26% because of excessive noise. Owing to the sensitivity to noise, the instance-based methods are not suitable for data mining using raw data. Although the prediction accuracy for NNge is extremely low, the TP rate of the 'Risky' class is relatively high compared to the results yielded using the other methods. Surprisingly, the prediction accuracy for the PRISM method is only 21.45%, but the TP rate for 'Declined' class is up to 0.977. The reason for the phenomenon of extremely low accuracy is the 'Declined' class of customers with apparent patterns or consistent characteristics. This unusual phenomenon raises the issues of identifying the patterns of 'Declined' and 'Risky' customers using the PRISM method. In fact, the particular algorithms such as

Table 3
Prediction performances using various methods of data mining for classification of two classes

Methods	Accuracy %	TP rate of		
		‘Declined’	‘Good’	MSE
Multilayer Perceptron	82.88	0.495	0.971	0.1383
Naı¨ve Bayes	82.85	0.493	0.971	0.1386
C4.5	82.80	0.486	0.974	0.1413
Voted Perceptron	82.77	0.49	0.971	0.1723
IBk	82.47	0.507	0.960	0.1400
ID3	82.37	0.509	0.960	0.1396
Conjunctive Rule	81.36	0.492	0.970	0.1504
Random Committee	80.29	0.521	0.521	0.1597
RBF Network	69.26	0.073	0.994	0.1995
PRISM	33.47	0.986	0.057	0.6654

PRISM can used as a filter that filters out specific instances in the training dataset. Inductively, the training dataset includes approximately 20% noise data, including contradiction and redundant instances, and data cleaning must be performed.

Experiment 13. To further examine data cleaning, this study postulates that the attribute of credit in the original training dataset only contains the two classes ‘Decline’ and ‘Good’, by merging the ‘Risky’ class into the ‘Decline’ one. The purpose of reducing the number of classes is to observe the variation in prediction accuracy. Conventionally, this investigation only attempts to demonstrate the well-known knowledge regarding which classifiers of two classes can gain higher prediction accuracy than others. Table 3 lists the performance parameters for the classification of three classes. By observing the experimental results in Table 3, the classifiers of two classes can only elevate the increment of accuracy of 1.3% on average except for the PRISM method. However, the PRISM method increases the prediction accuracy to 12%. This experimental result implies that performance can be further improved by manipulating data in certain matters. All of the matters are generally called data cleaning here. The study will focus on the combination of the method PRISM and the other method for the improvement of prediction accuracy. Incidentally, the Voted Perceptron may obtain a higher accuracy than IBk and ID3, but it can only be applied for implementing classifiers of two classes. In fact, regardless of the method applied, the difference of accuracy is very small, even when the number of predicted classes is reduced.

Experiment 14. From the above discussion, a combination of two methods may provide more knowledge regarding this real case. Particularly, the combination of a highly effective algorithm C4.5 (or ID3) and specific algorithm PRISM would be extremely interesting. Table 4 lists the results of data cleaning using a filter for classifying the prediction accuracy in advance. At the very beginning, a combination of the classification method C4.5 and the same filtering method as PRISM is tested. This filtering method is used to perform data cleaning of the training dataset for either removing or reserving specific instances from

Table 4
Results of data cleaning with a prior filter of classification for the prediction accuracy

Case	Classifier /filter TP rate for 'Declined', 'Risky', and 'Good'	Contents s[a]	Instance	Unclean accuracy (%)	Clean accuracy (%)
(a)	C4.5/C4.5	C2	8280	82.8	100
(b)	C4.5/C4.5	M2	1720	82.8	100
(c)	C4.5/C4.5	C3	8156	81.54	100
(d)	C4.5/C4.5	M3	1844	82.8	71.85
	(second pass)		515		99.42
(e)	ID3/C4.5	C3	8156	81.54	100
(f)	ID3/C4.5	M3	1844	81.54	69.03
	(second pass)		515		99.61
(g)	C4.5/Naı¨veBayes	M2	1720	82.85	99.71
(h)	Naı¨ve Bayes/ Naı¨ve Bayes	M2	1739	82.85	98.73
(i)	Naı¨ve Bayes /C4.5	M2	1739	82.8	99.02
(j)	Naı¨ve Bayes/ Naı¨ve Bayes	C2	8261	82.85	100
(k)	PRISM/PRISM	C3	2203	21.45	97.78
(l)	PRISM/PRISM	M3	7797	21.45	23.12
(m)	PRISM /C4.5	C3	8156	81.54	99.99
(n)	PRISM /C4.5	M3	1844	81.54	30.80
(o)	C4.5/PRISM 0.985, 0.543, 0.946	C3	2203	21.45	95.91
(p)	C4.5/PRISM 0, 0, 1	M3	7797	21.45	85.01
(q)	ID3/PRISM 0.987, 0.86, 0.969	C3	2203	21.45	97.05
(r)	ID3/PRISM 0, 0.02, 0.992	M3	7797	21.45	84.58

[a]Correctly classified for two classes: C2, misclassified for two classes: M2, correctly classified for three classes: C3, and misclassified for three classes: M3.

that dataset. Subsequently, various combinations of methods, such as C4.5, ID3, Naïve Bayes, and PRISM, are tested with either two or three classes. To facilitate explanation, the 'case' column in Table 4 uses labels from (a) to (q) to identify different cases. Moreover, the 'contents' column notes the type of cleansed content in the training dataset. The experiments include four different types, C2, M2, C3, and M3, which represent the correctly and incorrectly classified contents using filtering methods in situations involve two and three classes. For clarity of comparison, out of 10 000 instances, the resulting training datasets include the number of instances retained, and the prediction accuracy before the cleaning process is also listed to provide a cross reference. From observations of (a) and (b) in Table 4, the combination of the same methods using C4.5 can obtain a completely accurate classification for two classes. Despite correctly classified or misclassified instances, which respectively denote as prefix 'C' or 'M' in the column of Table 4, the C4.5 filter can identify the classification consistency for cases (a) and (b). This consistency is not surprising because of essentiality for decision tree. Notably, the so-called correctly classified instances or misclassified instances are gathered from the viewpoint of filtering, and then the instances are fed as input of the corresponding classifier. In fact, a decision-tree-based method is very suitable for use as a filter in data cleaning. In case (c), the classification using three classes using correctly classified instances is also similar to that in case (a). However, classification involving more than two classes raises the issue of contradiction between instances in the training dataset. Case (d) in the first pass attains an accuracy of only 71.85%, and needs to again filter the remaining instances in order to achieve 100% accuracy for those misclassified instances. The cause of contradiction is the classification with the three classes involving more noise than two classes. Multiple data cleaning can be used to eliminate the phenomenon of the classification involving instances of misclassification achieving low prediction accuracy. Correspondingly, cases (e) and (f) with C4.5 filter but ID3 classifier reveal the same situations as cases (c) and (d). Subsequently, cases (g), (h), and (i) combine the Naı¨ve Bayes method to be either the classifier or filter or even both. Consequently, the combination involving Naı¨ve Bayes method cannot achieve 100% accuracy even in two classes, because the essentialities of the decision tree are distinct from the Bayes rules. Case(j) may show this evidence obviously. Furthermore, because to the PRISM method prefers to reveal the rules with high regularities depending on the maximum probability, it can predict the class of ‘Declined’ very well. Therefore, in order to build a feasible learning model that all percentages of accuracy of prediction exceed 50% for every class one might need to involve the PRISM method into the combination of classifier/ filter. Case (k) contrasts with case (l); the misclassified instances by a PRISM filter are apparently unsuitable for being reclassified by a PRISM classifier. Similarly, cases (m) and (n) further confirm the other methods such as a filter of C4.5 and obtain the same observation result as cases (k) and (l). The observation result demonstrates that the instances misclassified using a filter of either C4.5 or PRISM thus does not conform to the properties modeled by PRISM. However, cases (o) and (p) reverse the order of combination of cases (m) and (n) to achieve reasonable accuracy. Particularly, case (q) using ID3 as a classifier achieves slightly better accuracy than using C4.5, and reaches an accuracy of 97.05%. Notably, the variation of preferred TP rate for each class transforms from the ‘Declined’ class into the ‘Good’ class as listed in Table 2. Besides, the most surprising result of data cleaning is shown in the case (q) marked with a gray background. Case (q) demonstrates a similarly high prediction accuracy for each class. This case applies the combination of classifier ID3 and filter PRISM to classification prediction, and achieves a very high TP rate for every class. Even case (r) using the misclassified instances as input also obtains an acceptable prediction accuracy. Additionally, the same experiment as case (q) applies to the training dataset that contains the same amount of instances for each class, and obtains the same conclusion (data not shown) as well. This analytical result confirms the conclusion explained in Experiment 9.

Experiment 15. Table 5 lists the prediction accuracy results using various treatments. Allof the cases are tested using the classification method C4.5. Cases (a), (b), and (c) in Table 5 remove each class once from the training dataset. These cases are equivalent to building a classifier with two classes. The rapid increase in accuracy after performing case (b) demonstrates that the ‘Risky’ class is difficult to predict. For each original attribute, the cases from (d) to (i) add a new attribute whose value is given by performing K-means clustering in advance and involves all attributes except this original attribute in conducting the

Table 5
Results of prediction accuracy using various treatments of experiments

Case	Treatments of experiments	Accuracy (%)
(a)	Remove instances with value ‘Good’	82.41
(b)	Remove instances with value ‘Risky’	93.33
(c)	Remove instances with value ‘Declined’	84.74
(d)	Add a clustering attribute ‘overdraft’	83.79
(e)	Add a clustering attribute ‘type’	83.65
(f)	Add a clustering attribute ‘transactions’	83.65
(g)	Add a clustering attribute ‘interest’	83.65
(h)	Add a clustering attribute ‘activity’	83.41
(i)	Add a clustering attribute ‘changed-signature’	83.65
(j)	Remove an attribute ‘overdraft’	72.08
(k)	Remove an attribute ‘type’	81.54
(l)	Remove an attribute ‘transactions’	80.97
(m)	Remove an attribute ‘interest’	81.54
(n)	Remove an attribute ‘activity’	81.54
(o)	Remove an attribute ‘changed-signature’	81.54
(p)	Reserve attributes ‘overdraft’, ‘type’, and ’transactions’	81.54
(q)	Reserve attributes ‘interest’, ‘activity’,’ and ‘changed-signature’	70.14
(r)	Remove with class value of ‘Risky’ and reserve attributes ‘overdraft’, ‘type’, and ’transactions’	93.34
(s)	Remove with class value of ‘Risky’ and reserve attributes ‘interest’, ‘activity’, and ‘changed-signature’	80.28
(t)	10% noise data applying for attribute ‘overdraft’	75.55
(u)	70% noise data applying for attribute ‘overdraft’	72.08
(v)	10% noise data applying for attribute ‘interest’	81.54
(w)	70% noise data applying for attribute ‘interest’	81.54
(x)	70% noise data applying for class attribute	33.54
(y)	Remove attribute ‘interest’, ‘activity’,’ ‘changed-signature’ and using the combination of ID3/PRISM for correctly classified instances	99.8 (1993 instances)
(z)	Remove attribute ‘interest’, ‘activity’,’ ‘changed-signature’ and using the combination of PRISM/C4.5 for correctly classified instances	99.99 (8156 instances)

classification. Consequently, the increment of accuracy is about 2%. The cases from (j) to (o) remove an attribute for each case. The decrease in accuracy resembles that in Experiment 7, which shows that the dominating attribute is indeed the attribute variable of ‘overdraft’. Additionally, cases (p) and (q) only reserve three attributes. The experimental result reveals that the set of attributes ‘overdraft’, ‘type’, and ’transactions’ can dominate the entire training model from the perspective of building a decision tree. Case (r) comprises cases (b) and (p) to clarify the relationship of instance-based and attribute-based data cleaning. The increment of accuracy adding case (p) approaches zero after applying case (b). Thus, the dominating attributes become important when the training dataset has been cleaned by case (b). However, the difference between cases (r) and (p) exceeds the difference of case

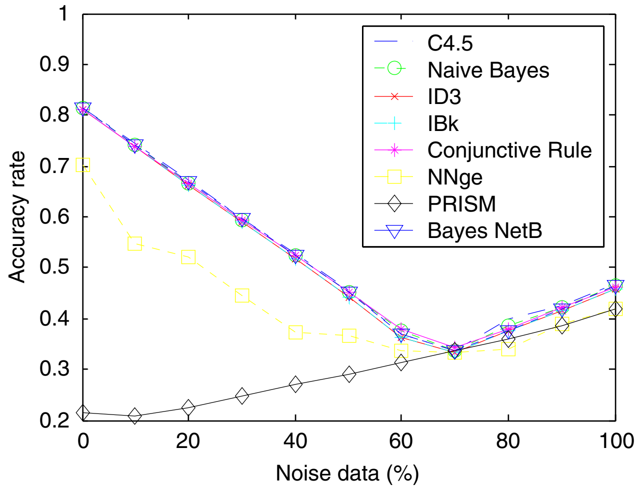

Fig. 13. Comparison of accuracy under noise interference for various methods.

(s) and (q) by around 2.2%. The expansion of differences implies that this class 'Risky' might affect the dominating attributes. Furthermore, cases from (t) to (x) apply a random noise of either 10% or 70% to the dominating, the non-dominating, and the class attribute. Comparing case (u) with case (x) revealed that the dominating attribute is better than the class attribute for the issue of noise interference. Fig. 13 displays the variation of accuracy in detail for various methods under noise interference. The different quantities of noise data accordingly and consistently affect the prediction accuracy variation in most methods except NNge and PRISM. Curiously, the addition of noise data gradually increases the prediction accuracy using the PRISM method. The curious phenomenon in the real case illustrates the particularity of PRISM, which may treat every class as the consistent training dataset for the classifier in equal proportion. Finally, cases (y) and (z) in Table 5 are compared with cases (q) and (m) in Table 4 to illustrate that classification with cleansed data may improve the prediction accuracy to 99.9%. The improvement of accuracy also simultaneously and fairly considers the TP rate for every class.

5. Conclusions

In this study, knowledge discovered by comparative analysis can facilitate the manipulation of skewed distribution data specific for the real-world example of checking accounts. Moreover, a series of classification experiments can supply sufficient evidence to explain random data behavior in real-world examples. However, regardless of the method used, the classification involving many contradictions, such as may occur in real cases, consistently obtains an accuracy of just 80%, and moreover, because to the number of good customers naturally significantly exceeds the number of bad customers. This study proposes two main methods of cleaning training datasets containing skewed distribution and contradictory data. One method involves randomly selecting instances in training dataset according to the values of class attributes so that the data distribution presents in equal proportion in the training dataset for each class. The other method involves building a combinative classifier using the combination of ID3 classifier and PRISM filter. Consequently, the proposed methods can obtain consistently high TP rates of prediction for each class. In fact, the 'Risky' class representing risky customers in a bank is the most interesting for bank managers in classification prediction. Thus, the study provides a very promising solution to discovering valuable business knowledge through data mining. Furthermore, this study also provides further insight into the financial field by determining which factors most influence classification accuracy for specific classes. The findings demonstrate that the classification accuracy for all classes is around 80% without data cleaning, but increases by up to 64 % for each class after cleaning the data and adjusting the proportion of training datasets. Additionally, the experiments also show that the 'overdraft' attribute dominates the average classification accuracy. Finally, an error of approximately 15% of the accuracy rate is tolerable. Accordingly, a bank manager can be granted a range of 15% authority to do credit scoring such as check loan. Moreover, numerous factors may affect customer credit, such as loans and other transaction information. Thus, such an error rate may be intolerable in other applications. This study suggests that applying the proposed solutions to the imbalance problem should be constrained to similar applications in the real-world. Finally, future studies should consider including more information in the preliminary model. However, not all experimental results were listed owing to space constraints. Future studies will explain the discovered knowledge from the perspective of risk management.

Acknowledgements

The authors would like to thank the Tainan Business Bank at the Republic of China for financially supporting this study.

References

[1] D. Aha, D. Kibler, Instance-based learning algorithms, Machine Learn. 6 (1991) 37–66.

[2] R.R. Bates, M. Sun, M.L. Scheuer, R.J. Sclabassi, 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.

[3] J. Cendrowska, PRISM: an algorithm for inducing modular rules, Int. J. Man-Machine Stud. 27 (1987) 349–370.

[4] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artificial Intell. Res. (JAIR) 16 (2002) 321–357.

[5] P. Domingos, Metacost: a general method for making classifiers cost-sensitive, in: Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, 1999, pp. 155–164.

[6] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01), 2001, pp. 973–978.

[7] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, The KDD process for extracting useful knowledge from volumes of data, Comm. ACM 39 (1996) 27–34.

[8] R. Fletcher, Practical Methods of Optimization, Wiley, New York, 1987.

[9] Y. Freund, R.E. Schapire, Large margin classification using the perceptron algorithm, in: Proceedings of the 11th Annual Conference on Computer Learning Theory, ACM Press, New York, 1998, pp. 209–217.

[10] R. Gerritsen, Assessing loan risks: a data mining case study, IEEE IT Professional (1999) 16–21.
[11] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, Ontario, Canada, 1999.
[12] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Machine Learn. 20 (1995) 197–243.
[13] C.M. Hung, Y.M. Huang, T.S. Chen, Assessing Check Credit with Skewed Data: A Knowledge Discovery Case Study, in: International Computer Symposium Workshop on Artificial Intelligence, Taiwan, December 2002.
[14] N. Japkowicz, S. Stephen, The class imbalance problem: a systematic study, Intell. Data Anal. 6 (5) (2002) 429–450.
[15] M. Jiang, X. Zhu, B. Yuan, X. Tang, B. Lin, Q. Ruan, M. Jiang, A fast hybrid algorithm of global optimization for feedforward neural networks, WCCC-ICSP International Conference on Signal Processing, vol. 3, 2000, pp. 1609–1612.
[16] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 1995, pp. 338–345.
[17] M. Kubat, S. Matwin, Addressing the Curse of Imbalanced Training Sets: One-Sided Selection, in: Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 179–186.
[18] C.X. Ling, C. Li, Data mining for direct marketing: problems and solutions, in: Proceedings of the Fourth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-98), ACM, New York, NY, 1998, pp. 73–79.
[19] M.J.D. Powell, Restart procedures for the conjugate gradient method, Math. Programm. 12 (1977) 241–254.
[20] M.J.D. Powell, Radial basis functions for multivariable interpolation: a review, in: RMCS, IMA Conference on Algorithms for the Approximation of Functions and Data, 1985, pp. 143–167.
[21] R. Pytlak, A globally convergent conjugate gradient algorithm, Proceedings of the 32nd IEEE Conference on Decision and Control, vol. 3, 1993, pp. 2890–2895.
[22] J.R. Quinlan, C.45: Programs for Machine Learning, Morgan Kaufmann, 1993.
[23] J.D. Rennie, L. Shih, J. Teevan, D. Karger, Tackling the Poor Assumptions of Naı¨ve Bayes Text Classifiers, in: ICML-2003, 2003, pp. 616–623.
[24] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychol. Rev. 65 (1958) 386–407 (Reprinted in Neurocomputing MIT Press, 1988).
[25] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations of back-propagation errors, Nature (London) 323 (1986) 533–536.
[26] O. Stan, E.W. Kamen, New block recursive MLP training algorithms using the Levenberg–Marquardt algorithm, IJCNN '99 International Joint Conference on Neural Networks, vol. 3, 1999, pp. 1672–1677.
[27] I.H. Witten, E. Frank, Data Mining practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, San Francisco, CA, 1999.
[28] B. Zadrozny, C. Elkan, Learning and making decisions when costs and probabilities are both unknown, in: Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, 2001, pp. 204–213.