Expert Systems with Applications 37 (2010) 1351–1360

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Support vector machine based multiagent ensemble learning for credit risk evaluation

Lean Yu [a,*], Wuyi Yue [b], Shouyang Wang [a], K.K. Lai [c]

[a] Institute of Systems Science, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
[b] Department of Intelligence and Informatics, Konan University, Kobe 658-8501, Japan
[c] Department of Management Sciences, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong

ARTICLE INFO

Keywords:
Multiagent ensemble learning
Support vector machine (SVM)
Diversity strategy
Ensemble strategy
Credit risk evaluation

ABSTRACT

In this paper, a four-stage support vector machine (SVM) based multiagent ensemble learning approach is proposed for credit risk evaluation. In the first stage, the initial dataset is divided into two independent subsets: training subset (in-sample data) and testing subset (out-of-sample data) for training and verification purposes. In the second stage, different SVM learning paradigms with much dissimilarity are constructed as intelligent agents for credit risk evaluation. In the third stage, multiple individual SVM agents are trained using training subsets and the corresponding evaluation results are also obtained. In the final stage, all individual results produced by multiple SVM agents in the previous stage are aggregated into an ensemble result. In particular, the impact of the diversity of individual intelligent agents on the generalization performance of the SVM-based multiagent ensemble learning system is examined and analyzed. For illustration, one corporate credit card application approval dataset is used to verify the effectiveness of the SVM-based multiagent ensemble learning system.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

The ongoing subprime mortgage crisis event originated from United States is attracting considerable concerns about credit risk analysis and evaluation. But in the previous studies, numerous methodologies, such as logit analysis (Wiginton, 1980), probit analysis (Grablowsky & Talley, 1981), *k*-nearest neighbor (KNN) (Henley & Hand, 1996), cluster analysis (Lim & Sohn, 2007), artificial neural networks (ANN) (Abdou, Pointon, & El-Masry, 2008; Malhotra & Malhotra, 2003; Smalz & Conrad, 1994), genetic algorithm (GA) (Varetto, 1998), evolutionary computation (Chen & Huang, 2003), support vector machine (SVM) (Bellotti & Crook, 2009; Chen & Shih, 2006; Huang, Chen, Hsu, Chen, & Wu, 2004; Lai, Yu, Zhou, & Wang, 2006c; Martens, Baesens, Van Gestel, & Vanthienen, 2007; Van Gestel, Baesens, Garcia, & Van Dijcke, 2003; Yu, Wang, & Lai, 2008), hybrid neural discriminant technique (Lee, Chiu, Lu, & Chen, 2002), neuro-fuzzy (Piramuthu, 1999), hybrid SVM technique (Chen, Ma, & Ma, 2009), neural network ensemble model (Lai, Yu, Wang, & Zhou, 2006b; Yu et al., 2008), neural network metalearning technique (Lai, Yu, Zhou, & Wang, 2006d), SVM metamodel (Lai, Yu, Huang, & Wang, 2006a), fuzzy SVM (Wang, Wang, & Lai, 2005), fuzzy group decision making (GDM) model (Yu, Wang, & Lai, 2007) were proposed for this important issue and thus making credit risk evaluation become a flourishing research stream (Elliott & Filinkov, 2008; Kim & Sohn, 2004; Lin, 2009; Mues, Baesens, Files, & Vanthienen, 2004; Quah & Sriganesh, 2008; Thomas, 2002; Thomas, Edelman, Crook, 2002; Thomas, Oliver, Hand, 2005). With the advancement of computer technologies and computational techniques, this research stream has gained more momentum than before. Recent studies (e.g. Chen, 2007; Lai et al., 2006b; Yu et al., 2008; Yu, Wang, Lai, & Zhou, 2008) have discovered that the multiagent learning system is an efficient way to achieve high performance, especially in fields where the development of a powerful single learning system requires considerable effort (Yu et al., 2008).

Generally, multiagent learning can be divided into two categories: competitive learning, where agents work asynchronously on the same problem and the result of the best agent is the final output result, and cooperative learning, where the final output result is a fusion or aggregation of the individual results of some agents. However, past studies have revealed that an effective multiagent learning system may not be an individual model by a single learning agent, but the combination or ensemble of some agents (Olmeda & Fernandez, 1997; Yu, Wang, & Lai, 2005). Note that in the context of multiagent learning systems, an agent is usually defined as an independent learning unit participating as a member of the multiagent learning system. Usually, ensemble learning models outperform single learning models, whose performance is limited by the imperfection of feature extraction, learning algorithms,

* Corresponding author. Tel.: +86 10 62565817; fax: +86 10 62541823.
E-mail address: yulean@amss.ac.cn (L. Yu).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.06.083

and the inadequacy of training data. Another reason supporting this proposition is that different single learning models have their inherent drawbacks. Aggregating them may thus lead to a better model with a high generalization capability. From the above descriptions, we can conclude that there are two essential requirements in a multiagent ensemble learning system. The first is that the ensemble members or learning agents must be diverse or complementary, i.e., agents must exhibit different properties. Another condition is that an optimal ensemble strategy is also required to fuse a set of diverse agents (Yu et al., 2007, 2008; Chen, 2007; Yu et al., 2008).

To achieve high performance, this paper attempts to utilize a highly competitive machine learning tool – support vector machine (SVM), first proposed by Vapnik (1995) – as a generic ensemble learning agent for credit risk evalution. The main reasons of selecting SVM as an ensemble learning agent are three-fold. First of all, SVM requires less prior assumptions about the input data, such as normal distribution and continuity. This is different from traditional statistical models. Second, SVM can perform a nonlinear mapping from an original input space into a high dimensional feature space, in which it constructs a linear discriminant function to replace the nonlinear function in the original low-dimension input space. This characteristic also solves any dimension disaster problem because its computational complexity is not dependent on the sample dimension. Third, SVM attempts to learn the separating hyperplane to maximize the margin, therefore implementing structural risk minimization (SRM) and realizing good generalization capability. This pattern can directly help SVM escape local minima, which often occurs in the training of ANN. These important characteristics will also make SVM popular in many practical application problems (Vapnik, 1995; Yu et al., 2008).

The basic procedure of using SVM as a generic ensemble learning agent to construct a multiagent learning system consists of four stages. In the first stage, an initial dataset is divided into two independent subsets: training subset and testing subset. In some necessary situations, the third subset, validation subset may be produced to overcome overfitting problem. In the second stage, diverse SVM models are created to formulate the generic ensemble learning agents (i.e., SVM agents) through using some diversity strategies. In the third stage, the diverse SVM models are trained by the training subset to produce different results. In the final stage, these different SVM agents are integrated into an aggregated output using a specific ensemble strategy.

However, in the process of developing an effective multiagent ensemble learning system, two generic prerequisites: constructing diverse learning agents with much disagreement and adopting optimal ensemble strategies, should be carefully considered to improve the generalization performance, as previously mentioned. For these purposes, different diversity strategies and different ensemble strategies will be investigated in this paper.

On the one hand, the past literature review found that a key to the performance improvement of a multiagent ensemble learning system is the diversity strategies of learning agents in an ensemble learning process. If each learning agent in the ensemble system thinks alike, then they will come to the same conclusion and thus there may be no performance improvement. In order to investigate the impact of diversity on the final generalization of the multiagent ensemble learning system, three typical diversity strategies, data diversity, parameter diversity, and kernel diversity, are investigated in a SVM-based multiagent ensemble learning system.

On the other hand, a core step of constructing an effective multiagent ensemble learning system is to integrate the individual results produced by single learning agents into an aggregated output. Similar to the previous descriptions, there are two main strategies: competitive ensemble strategy and cooperative ensemble strategy. Essentially, the result of competitive ensemble is an individual learning system with a single best agent. But the problem lies in that the selected best model cannot guarantee its superiority in all cases and all scenarios. In this sense, cooperative ensemble strategy is often preferable in most practical applications. In the past studies, some different methods in cooperative ensemble strategy have been examined. Typically, majority voting and weighted averaging have been commonly used ensemble approaches in a multiagent ensemble learning system. Particularly, in such approaches, there are many variants in weighted averaging. These approaches will be investigated in the following sections.

The main aims of this paper are to construct a SVM-based multiagent ensemble learning system for credit risk evaluation and to investigate the effect of both diversity strategy and ensemble strategy on the generalization performance of the SVM-based multiagent ensemble learning system. The rest of this paper is organized as follows. The next section presents, in detail, a formulation process of a multi-stage SVM-based multiagent ensemble learning system. For illustration and verification purposes, a practical credit card application approval example is used and the corresponding computational results are reported in Section 3. Section 4 concludes this paper.

2. Methodology formulation

In this section, a four-stage SVM-based multiagent ensemble learning approach is proposed for credit risk evaluation problems. First of all, the initial dataset is divided into two independent subsets: training subset (in-sample data) and testing subsets (out-of-sample data) for learning and testing purposes. A validation subset is also included in the training subset by *k*-fold cross validation approach. Then, diverse SVM learning paradigms are used as intelligent agents to analyze and evaluate credit risk. Subsequently, multiple individual SVM agents are trained using training subset and the corresponding evaluation results are also obtained. Finally, all individual results produced by multiple single SVM agents in the previous stage are aggregated into a final output result.

2.1. Data partition

In applications of SVM on multiagent ensemble learning, data partition is a necessary step. Some research results (e.g., Dawson & Wilby, 1998; Maier & Dandy, 2000) shown that the data division can have a significant impact on the results obtained. In previous similar studies, data division was carried out in all cases. Generally, data partition is carried out on an arbitrary basis, and the statistical properties of the respective datasets are seldom considered. In most cases, two subsets, i.e., training subset and testing subset, are used. In this paper, the overall dataset is also divided into two subsets. Eighty percent of the data are used as the training subset and the remainder is used as the testing subset. Note that *k*-fold cross validation is used in the paper to achieve a more reliable result.

2.2. Diverse SVM agent creation

In order to capture the implicit patterns hidden in the dataset from different perspectives, diverse SVM agents should be used. Generally, an effective multiagent ensemble learning system consisting of diverse models with much disagreement is more likely to have a good generalization performance in terms of the principle of bias-variance trade-off (Yu, Lai, Wang, & Huang, 2006). Therefore, how to generate the diverse models is a crucial factor for constructing an effective multiagent ensemble learning system. For

SVM models, several diversity strategies have been investigated for the generation of ensemble members making different errors. Such diversity strategies basically relied on varying the training samples and parameters related to the SVM design. In particular, some main diversity strategies can be categorized into the following three aspects:

2.2.1. Data diversity

Because different data often contain different information, these different data can generate diverse models with dissimilarity. Usually, in intelligent learning models, training dataset is used to construct a concrete model and thus different training dataset will be an important source of diversity if the training dataset at hand is functionally or structurally divisible into some distinct training subsets. In order to achieve diverse models, some typical data sampling approaches, such as bootstrap aggregating (bagging) proposed by Breiman (1996) and noise injection (Yang & Browne, 2004), are used to create diversity. In this paper, the bagging algorithm is adopted as a data sampling tool.

The bagging approach is a widely used data sampling method in machine learning. Given that the size of the original data set *DS* is *P*, the size of new training data is *N*, and the number of new training data points is *m*, that is, each new training subset *TR* has *m* data points, $TR_1, TR_2, \ldots, TR_m$. Accordingly the bagging algorithm for generating new training subsets can be shown in Fig. 1.

The bagging algorithm is very efficient in constructing a reasonably sized training subset when the size of the original data set is small due to the feature of its random sampling with replacement. Therefore, the bagging is a useful data sampling method for machine learning (Breiman, 1996). In this paper, we use the bagging algorithm to generate different training subsets.

2.2.2. Parameter diversity

In an SVM model, there are two classes of typical parameters: regulation parameter and kernel parameters. By changing the SVM model parameters, different SVM models with high levels of disagreement can be produced. In this paper, we will investigate the impacts of different kernel parameters on SVM generalization performance.

2.2.3. Kernel diversity

In an SVM model, the kernel function has an important effect on the generalization performance of SVM. Hence, using different kernel functions in the SVM models can also create diverse SVM models. In the SVM model, the polynomial function, Sigmoidal function, and RBF function, are several typical kernel functions. Similarly, this paper will also examine the influences of different kernel functions on the SVM generalization performance.

Input: original data set *DS*
Output: The generated new training subsets (TR_1, TR_2, …, TR_m)
For $t = 1$ **to** m
  For $i = 1$ **to** N
    RandRow = *P* * *rand* ()
    If *RandRow* <= *P*
      $P_t(i, AllColumns) = DS(RandRow, AllColumns)$
    End If
  Next *i*
Next *t*
Output the final training subsets (TR_1, TR_2, …, TR_m)

Fig. 1. The bagging algorithm.

2.3. Single SVM agent learning

After determining diverse SVM models, the next step is to train the different SVM agents to produce different output results using training subsets. In our paper, the standard SVM proposed by Vapnik (1995) is used as the intelligent learning agent.

The generic idea of SVM is to maximize the margin hyperplane in the feature space. Similar to other supervised learning methods, an underlying theme of the SVM is to learn from data, which adopts the structural risk minimization (SRM) principle from computational learning theory. Usually, SVM can be used as regression and classification. In this paper, we focus on the classification problem.

Let x be an input vector as $x = \{x_1, x_2, \ldots, x_N\}$, where x_i ($i = 1,2,\ldots,N$) is the ith element of x in the training subset. Let y be an output vector as $y = \{y_1, y_2, \ldots, y_N\}$, where y_i ($i = 1,2,\ldots,N$) is the ith element of y. N is the number of training data points. In SVM classification, y_i is the class value of the training data point x_i. Using x_i and y_i, the SVM classification problem can be represented in the following optimization problem:

$$\begin{cases} \min & J(a,b,\xi) = (1/2)a^T a + C\sum_{i=1}^{N}\xi_i, \\ \text{s.t.} & y_i[\varphi(x_i)\cdot a_i + b] \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1,2,\ldots,N, \end{cases} \tag{1}$$

where a is a normal vector of the hyperplane as $a = \{a_1, a_2, \ldots, a_N\}$, where a_i ($i = 1,2,\ldots,N$) is the ith element of a. b is a bias that is a scalar. $\phi(x_i)$ is a nonlinear mapping function for the ith input data x_i from a low-dimension space to a high-dimension space. ξ is a tolerable misclassification error vector as $\xi = \{\xi_1, \xi_2, \ldots, \xi_N\}$, where ξ_i ($i = 1,2,\ldots,N$ is the ith misclassification error of ξ. C is a regulation parameter controlling the trade-off between the maximal margin and the tolerable misclassification errors. When C is large, the error term will be emphasized. Small C means that the large classification margin is encouraged.

Vapnik (1995) found that training a SVM model will lead to a quadratic programming (QP) problem with bound constraints and linear equality constraints, as shown in Eq. (2).

$$\begin{cases} \max & J(\alpha) = \sum_{i=1}^{N}\alpha_i - (1/2)\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \varphi(x_i)^T\varphi(x_j), \\ & \quad = \sum_{i=1}^{N}\alpha_i - (1/2)\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j), \\ \text{s.t.} & \sum_{i=1}^{N}\alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, i = 1,2,\ldots,N, \end{cases} \tag{2}$$

where α is a vector of Lagrange multipliers as $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$, where α_i ($i = 1,2,\ldots,N$) is the ith Lagrange multiplier of α corresponding to ith inequality constraint defined in Eq. (1). $K(x_i,x_j)$ ($i,j = 1,2,\ldots,N, i \neq j$) is defined as the kernel function with $K(x_i,x_j) = \phi(x_i)^T\phi(x_j)$. The elegance of using the kernel function is that one can deal with feature spaces of arbitrary dimensionality without having to compute the nonlinear map function $\phi(x)$ explicitly, where $x = \{x_1, x_2, \ldots, x_N\}$. Any function that satisfies Mercer's condition (Vapnik, 1995) can be used as the kernel function for SVM models. Typical examples of the kernel function are the polynomial kernel $K_{poly}(x_i,x_j) = (x_i^T x_j + 1)^d$, radial basis function (RBF) kernel: $K_{rbf}(x_i,x_j) = \exp(-|\ x_i - x_j|^2/\sigma^2)$, and sigmoid kernel $K_{sig}(x_i,x_j) = \tanh(\rho x_i^T x_j + \theta)$, where d, σ, ρ, and θ are the kernel parameters.

From the implementation point of view, training SVM is actually equivalent to solving the linearly constrained QP problem. By solving the QP problem, the final output function $f(z)$ of the SVM model can be represented as

$$f(z) = \sum_{l=1}^{N_L} \alpha_l^{(r)} y_l^{(r)} K(z, x_l^{(r)}) + b, \tag{3}$$

where $\alpha_l^{(r)} (l = 1, 2, \ldots, N_L)$ is the lth element of a support vector $\alpha^{(r)}, \alpha^{(r)} = \left\{\alpha_1^{(r)}, \alpha_2^{(r)}, \ldots, \alpha_{N_L}^{(r)}\right\}$, which is selected from α, the vector of Lagrange multipliers, defined in Eq. (2). N_L is the number of elements of the support vector $\alpha^{(r)}$. Generally, the N_L is less than the number of training data points N in many practical problems, namely, $N_L \leq N$. $x_l^{(r)} (l = 1, 2, \ldots, N_L)$ is the lth element of a vector $x^{(r)}$ corresponding to the lth element of support vector $\alpha^{(r)}$. The vector $x^{(r)}$ is a part of the input vector x as $x^{(r)} = \left\{x_1^{(r)}, x_2^{(r)}, \ldots, x_{N_L}^{(r)}\right\}$ selected from the input vector x satisfying the conditions shown in Eq. (2). $y_l^{(r)} (l = 1, 2, \ldots, N_L)$ is the lth element of $y^{(r)}$ corresponding to the lth element of support vector $\alpha^{(r)}$. The vector $y^{(r)}$ is a part of output vector y as $y^{(r)} = \left\{y_1^{(r)}, y_2^{(r)}, \ldots, y_{N_L}^{(r)}\right\}$ selected from the output vector y satisfying the conditions shown in Eq. (2). z is a testing data vector as $z = \{z_1, z_2, \ldots, z_m\}$, where m is the number of testing data points and b is defined in Eq. (1).

Through training SVM, model parameters can be determined and accordingly the SVM classifier $F(z)$ can be represented as

$$F(z) = sign(f(z)) = sign\left(\sum_{l=1}^{N_L} \alpha_l^{(r)} y_l^{(r)} K(z, x_l^{(r)}) + b\right), \tag{4}$$

where *sign* is an indicator function and other symbols are identical to Eq. (3).

With the above SVM classifier, some results for a specific problem can be achieved.

2.4. Multiagent ensemble learning

When individual SVM agents are generated, each SVM agent can output its own computational results for a specific problem. Although one agent based on a single SVM model may have good performance, it is sensitive to samples and parameter settings, i.e. each single SVM agent may have some biases. One effective way to reduce the bias is to integrate these SVM agents into an aggregated output for final results. Fig. 2 shows the structure of the multiagent ensemble learning system, where p is the number of learning agents.

From the previous descriptions and Fig. 2, it is easy to find that how to construct the combiner, and how to create diversity among the multiple agents, are the two main factors in building an effective multiagent ensemble learning system. Because the previous sections have introduced some diverse SVM agents by using different diversity strategies, how to construct an efficient combiner has been a key factor in constructing an effective multiagent ensemble learning system.

However, before integrating these SVM agents, strategies for selecting SVM agents must be noted. Generally, these strategies can be divided into two categories: (i) producing an exact number of SVM learning agents; and (ii) overproducing SVM learning agents and then selecting a subset of these overproduced SVM agents (Yang & Browne, 2004).

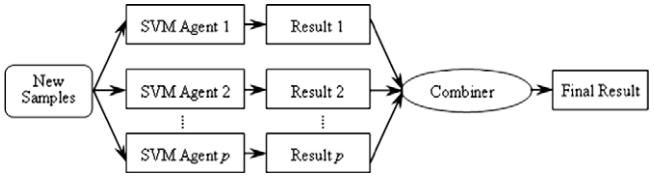

Fig. 2. The structure of multiagent ensemble learning system.

In the first strategy, several common ensemble approaches, e.g., boosting (Schapire, 1990), can be employed to generate the exact number of diverse learning agents for ensemble purpose. That is, no selection process will be used in this strategy, and all the learning agents produced will be combined into an aggregated output. In the second strategy, the main aim is to create a large number of agent candidates and then choose some most diverse agents for the ensemble. The selection criterion includes some error diversity measures, such as mean squares errors (MSE), which is introduced in detail by Partridge & Yates (1996). Because the first strategy is based upon the idea of creating diverse SVMs at the early stage of ensemble design, it is more effective than the second strategy; especially for some conditions where some powerful computing resources are restricted. However, the main reason is that the second strategy is not preferred is because it cannot avoid occupying large amounts of computing time and storage space while creating a large number of ensemble agent candidates, some of which are to be later discarded (Yang & Browne, 2004).

Generally, there are some ensemble strategies to take into account different results in the existing literatures. Typically, majority voting and weighted averaging are two popular ensemble strategies. The main objective of majority voting strategy is to determine the vote of the majority of the population of learning agents. To our knowledge, majority voting is one of the most widely used ensemble strategies due to its easy implementation. In the majority voting strategy, each agent has the same weight and the voting of the ensemble agents will determine the final result. Usually, it takes over half the ensemble agents to agree in order for a result to be accepted as the final output of the ensemble, regardless of the diversity and accuracy of each agent's generalization. In the mathematic form, the result of the final ensemble output $G(z)$ can be represented as

$$G(z) = sign\left(\sum_{k=1}^{p} F_k(z)/p\right), \tag{5}$$

where z is a testing data vector as $z = \{z_1, z_2, \ldots, z_m\}$ defined in Eq. (3), $F_k(z)$ is the output result of the kth learning agent presented in Eq. (4), using the testing data vector z, m is the number of testing data points and p is the number of learning agents. Although this strategy is easy to be used and implemented, a serious problem associated with majority voting is that it ignores the fact that some learning agents that lie in a minority sometimes produce the more accurate results. At the ensemble stage, it ignores the existence of diversity that is the motivation for a multiagent ensemble learning system (Yang & Browne, 2004).

Weighted averaging is where the final output result is calculated in terms of single agents' performance, and a weight is attached to each single agent's output. The sum of the total weight is one and each agent is entitled to a portion of this total weight based on their performance. A typical binary classification example is to use validation examples to determine the performance weight. Suppose that AB represents the number of observed Class A instances that are mistakenly classified as Class B by an agent, AA denotes the number of correctly classified Class A instances that belong to Class A; BA represents the number of observed Class B instances that are mistakenly classified as Class A, while BB represents the number of correctly classified Class B instances that belong to Class B. Some common measures used to evaluate the performance of the agent are defined below:

$$\text{Type I Accuracy} = \text{Specificity} = \frac{BB}{BB + BA}, \tag{6}$$

$$\text{Type II Accuracy} = \text{Sensitivity} = \frac{AA}{AA + AB}, \tag{7}$$

$$\text{Total Accuracy (TA)} = \frac{AA + BB}{AA + AB + BB + BA}. \tag{8}$$

Each SVM agent is trained by a training dataset and is verified by the validation samples. Assume that the total accuracy of validation samples of agent k is denoted by TA_k, the weight of SVM agent k denoted by w_k can be calculated as

$$w_k = \frac{TA_k}{\sum_{k=1}^{p} TA_k}. \quad (9)$$

Then the final ensemble output value $G(z)$ of this strategy is shown as follows:

$$G(z) = sign\left(\sum_{k=1}^{p} w_k \cdot F_k(z)\right), \quad (10)$$

where w_k is the kth weight for kth agent and other symbols are identical to Eq. (5).

The above *TA*-based weight averaging ensemble strategy is a typical class of ensemble strategy, the determination of the weight is heavily dependant on the validation samples. When there is no validation subset in data division, this ensemble strategy will be useless. For this reason, this paper proposes an adaptive weight determination method based on an adaptive linear neural network (ALNN) model.

ALNN is a single-layer neural network, where the transfer function is a pure linear function, and the learning rule is Widrow-Hoff (i.e. least mean squared (LMS) rule) (Hagan, Demuth, & Beale, 1996). Applying the mathematical form presented in the above referencing book, the final ensemble output value $G(z)$ can be expressed as follows:

$$G(z) = \phi\left(\sum_{k=1}^{p} s_k F_k(z) + h_k\right), \quad (11)$$

where s_k is the kth weight of ALNN attached to the result of the kth agent, and h_k is the kth bias of ALNN, $k = 1,2,\ldots,p$. $\phi()$ is a transfer function of the single-layer ALNN, $F_k(z)$ and z are defined in Eq. (3). Usually, the LMS rule adjusts the weights and bias so as to minimize the mean square error (MSE) as follows:

$$\mathrm{MSE} = \frac{1}{p}\sum_{k=1}^{p}(e_k)^2 = \frac{1}{p}\sum_{k=1}^{p}(u_k - v_k)^2, \quad (12)$$

where $e_k = u_k - v_k$. e_k is the kth element of an error vector e as $e = \{e_1, e_2, \ldots, e_p\}$, u_k is the kth element of the target output vector u as $u = \{u_1, u_2, \ldots, u_p\}$, and v_k is the kth element of the neural network output vector v as $v = \{v_1, v_2, \ldots, v_p\}$. p is the number of the learning agents. The change to the network weight s_k and the bias h_k for multiple neurons can be written as follows:

$$s_{k+1} = s_k + 2\eta e_k u_k, \quad (13)$$

$$h_{k+1} = h_k + 2\eta e_k, \quad (14)$$

where η is a learning rate that is a scalar. To obtain a good convergence to the optimal weight s_k^* and bias h_k^*, the learning rate must be less than the reciprocal of the largest eigen value of the correlation matrix $z^T z$ of the testing data vectors $z = \{z_1, z_2, \ldots, z_m\}$.

In this paper, the outputs of all single SVM agents are first used as the input vector of the ALNN. Then the ALNN is trained with the LMS algorithm and corresponding weights are determined adaptively. Accordingly, the final results are also produced by the ALNN in the form of Eq. (10).

2.5. Overall process of the proposed multiagent ensemble learning system

According to the previous descriptions, the overall process of the proposed multiagent ensemble learning system can be summarized into the following four stages.

Stage
I: Data Division
An original dataset is divided into several non-overlapping subsets for learning and testing, i.e., training subset and testing subset.
Stage
II: Diverse SVM Agent Creation
To construct an effective multiagent ensemble learning system, multiple diverse SVM models with high levels of disagreement are used as intelligent agents for modeling purpose.
Stage
III: Single SVM Agent Learning
When a single SVM agent is produced, these agents can be trained with the use of a training subset obtained from Stage I. Accordingly, the single results can be obtained.
Stage
IV: Multiagent Ensemble Learning
After the generalization results of individual agents are produced, an adaptive linear neural network (ALNN) is used to fuse the single output results of individual learning agents into an aggregated output result.

In order to verify the effectiveness of the proposed SVM-based multiagent ensemble learning system, a real-world credit card approval experiment is conducted and analyzed.

3. Data description and experiment design

In this section, a British credit card application approval dataset is used. It is from a financial service company in England, obtained from the accessory CDROM of Thomas, Edelman, Crook (2002). The dataset includes detailed information of 1225 applicants, including 323 observed bad applicants. In the 1225 applicants, the number of good cases (902) is nearly three times that of bad cases (323). To make the numbers of the two classes near equal, we triple the number of bad cases, i.e. we add two copies of each bad case. Thus the total dataset grows to 1871 cases. The purpose of doing this is to avoid having too many good cases or too few bad cases in the training samples. Then we randomly draw 1000 cases comprising 500 good cases and 500 bad cases from the total of 1871 cases as the training samples and treat the rest as testing samples (i.e. 402 good applicants and 469 bad applicants). In these cases, each case is characterized by 14 decision attributes, which are described as follows:

(1) Year of birth.
(2) Number of children.
(3) Number of other dependents.
(4) Is there a home phone.
(5) Applicant's income.
(6) Applicant's employment status.
(7) Spouse's income.
(8) Residential status.
(9) Value of home.
(10) Mortgage balance outstanding.
(11) Outgoings on mortgage or rent.
(12) Outgoings on loans.
(13) Outgoings on hire purchase.
(14) Outgoings on credit cards.

Using this dataset and previous descriptions, we can construct a practical SVM-based multiagent ensemble learning system for credit risk evaluation. The basic purpose of the multiagent ensemble learning model is to make full use of knowledge and intelligence of each agent in the multiagent ensemble learning system. In this multiagent ensemble learning system, agents are some intelligent techniques or intelligent agents.

Subsequently, in order to investigate the impacts of diversity strategy and ensemble strategy on the performance of a multiagent ensemble learning system, we perform different experiments with diverse diversity strategies and ensemble strategies. When testing the effects of diversity strategies on the performance of a multiagent ensemble learning system, the ensemble strategy is assumed to be fixed. Similarly, the diversity strategy will be fixed if the impacts of ensemble strategies on the performance of the multiagent ensemble learning system are evaluated.

In the testing of diversity strategies, a majority voting based ensemble strategy is adopted. For data diversity, a bagging algorithm is utilized. In this paper, we use five different training scenarios for SVM learning. For example, we use 500 training sets (i.e. $P = 1871$, $N = 1000$, and $m = 500$) to create 500 different evaluation results for each intelligent agent. In this case, the SVM model with a RBF kernel is used as an intelligent learning agent. The parameters of SVM including regulation parameter and kernel parameters are determined by a grid search.

For parameter diversity, we use SVM models with the RBF kernel as intelligent learning agents. Accordingly, the regulation parameter C and RBF kernel parameter σ^2 will be changed. The training data used is the initial training data produced by data partition.

For kernel diversity, three typical kernel functions, polynomial function, RBF function and sigmoid function are used. In this case, the parameters of SVM are determined by a grid search. The training data used is the initial training data produced by data partition.

In the testing of ensemble strategies, bagging-based diversity strategy is used. In this case, the SVM models with the RBF kernel are used as intelligent learning agents. The parameters of SVM are determined by a grid search.

To guarantee the robustness of the SVM model, the two-fold cross validation (CV) method is used. In the two-fold CV method, the first step is to divide the training dataset into two non-overlapping subsets. Then we train a SVM using the first subset and validate the trained SVM with the second subset. Subsequently, the second subset is used for training and the first subset is used for validation. Use of the two-fold CV is actually a reasonable compromise, considering the computational complexity (i.e., Quadratic Programming problem) of the SVM learning paradigms. Furthermore, an estimate from the two-fold CV is likely to be more reliable than an estimate from a common practice using a single validation set. Besides model robustness, the validation sample of this experiment is also used to find weights of different agents for ensemble purpose, as mentioned in Section 2.4.

In addition, for comparative purpose, five popular classification models: linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logit regression (LogR), feed-forward neural network (FNN) and support vector machine (SVM), as well as three multiagent learning models: majority voting based (MV) multiagent ensemble learning model, TA-based (TA) weight averaging multiagent ensemble learning model and ALNN-based (ALNN) multiagent ensemble learning model are utilized in experiments. The final specification is that the classification accuracy in the testing set is used as the performance evaluation criterion. Typically, three evaluation criteria, Type I, Type II and Total Accuracy, as defined in Section 2.4, are used to measure the classification performance.

4. Experimental results

Using the previous experiment design, three types of different experiments were conducted. The first was the diversity strategy testing experiment, the second was the ensemble strategy testing experiment, and the final one was the performance comparison of different models.

4.1. Diversity strategy testing

In this subsection, three different diversity strategies, data diversity, parameter diversity and kernel diversity, were used to investigate the impact on the multiagent ensemble learning model in group decision making.

In the data diversity testing, the main goal was to examine the impact of the number of training samples on the generalization performance of the multiagent ensemble learning model. For this purpose, we adopted the bagging algorithm to create 100, 500, 1000, 2000 and 5000 different training samples (5 different scenarios), as mentioned previously. Using these different training samples, different single SVM models with dissimilarities were produced. With the trained SVM models, some classification results were achieved. Using the majority voting strategy, the final multiagent ensemble results were obtained from the multiagent ensemble learning system. Accordingly the final computational results are shown in Table 1. Note that the values in brackets are standard deviations.

Based on the results in Table 1, two conclusions can be easily drawn. On the one hand, as the number of training samples increases, the performance improvements are increased from the perspective of total accuracy. On the other hand, the degree of performance improvement from 100 datasets to 1000 datasets is larger than the degree of performance improvement from 1000 datasets to 5000 datasets, which indicates that the degree of performance improvement is not fully dependant on the number of training samples.

In the parameter diversity testing, the SVM models with the RBF kernel are used as intelligent learning agents, as specified in Section 3. Therefore, different regulation parameters C and kernel parameters σ^2 are used to create diversity. In particular, the regulation parameter C varies from 10 to 100 with a step size of 10 and the kernel parameter σ^2 changes from 10 to 100 with a step size of 10. Some computational results are shown in Table 2.

From Table 2, we can observe that different parameter settings can lead to different generalization performance, but the differences are insignificant. More specifically, the classification performance on the testing data increases when C increases from 10 to 80, but the classification performance decreases when C increases from 90 to 100. Thus a suitable regulation parameter is of utmost importance to SVM learning. The main reason is that the regulation parameter C is an important trade-off parameter between margin maximization and tolerable classification errors. If the selection of regulation parameter C is inappropriate, it might lead to unexpected generalization results. But these results partly support the conclusions of Kim (2003) & Tay & Cao (2001). However, for kernel parameter σ^2, it is difficult to find similar results. The possible reason is worth exploring further in the future research.

Similarly, we can also use different kernel functions to create a multiagent SVM ensemble learning system for credit risk evaluation. For this purpose, three different kernel functions: polynomial function, sigmoid function and RBF function are used for testing. In

Table 1
Performance comparisons of SVM ensemble learning with data diversity.

Number of training samples	Type I (%)	Type II (%)	Total accuracy (%)
100	66.94 [3.58]	63.45 [3.93]	65.12 [3.77]
500	69.38 [3.23]	66.36 [3.71]	67.82 [3.52]
1000	73.44 [2.81]	67.93 [3.26]	70.54 [3.03]
2000	73.97 [2.76]	68.14 [3.15]	70.91 [2.87]
5000	73.65 [2.73]	68.72 [3.29]	71.06 [2.85]

Table 2
Performance comparisons of SVM ensemble learning with different parameters.

C	σ^2	Type I (%)	Type II (%)	Total accuracy (%)
10	10, 20,...,100	67.48 [4.01]	63.75 [3.54]	65.32 [3.82]
20	10, 20,...,100	69.34 [3.84]	64.89 [3.78]	66.81 [3.94]
30	10, 20,...,100	68.96 [3.33]	65.05 [3.66]	66.88 [3.53]
40	10, 20,...,100	69.58 [4.21]	64.98 [4.08]	67.14 [4.14]
50	10, 20,...,100	70.24 [3.57]	65.46 [4.14]	67.51 [3.88]
60	10, 20,...,100	69.87 [4.01]	66.63 [4.26]	68.22 [4.12]
70	10, 20,...,100	71.32 [3.82]	67.89 [4.01]	69.44 [3.96]
80	10, 20,...,100	70.45 [3.51]	69.34 [3.83]	69.53 [3.73]
90	10, 20,...,100	70.08 [4.14]	69.08 [4.54]	69.39 [4.31]
100	10, 20,...,100	69.87 [4.76]	67.85 [5.04]	68.57 [4.89]
10, 20,...,100	10	69.65 [4.25]	67.42 [3.96]	68.93 [4.15]
10, 20,...,100	20	71.93 [3.92]	66.38 [4.33]	68.67 [4.18]
10, 20,...,100	30	68.02 [4.61]	66.42 [3.89]	67.14 [4.27]
10, 20,...,100	40	69.98 [4.83]	67.02 [4.22]	68.49 [4.54]
10, 20,...,100	50	68.69 [3.95]	64.87 [4.45]	66.78 [4.29]
10, 20,...,100	60	69.65 [3.32]	66.22 [3.86]	67.54 [3.66]
10, 20,...,100	70	67.43 [4.47]	66.38 [3.92]	66.76 [4.23]
10, 20,...,100	80	68.52 [4.81]	65.15 [3.74]	66.82 [4.43]
10, 20,...,100	90	66.49 [4.09]	65.41 [4.45]	65.78 [6.09]
10, 20,...,100	100	67.56 [3.85]	66.24 [4.08]	66.78 [3.97]

order to construct a diverse ensemble learning system, 1000 copies of the same training dataset are replicated. In the 1000 training data points, different kernel functions are hybridized to create the kernel diversity. Note that in every experiment only one kernel function dominates the multiagent ensemble learning system. Detailed configuration of the kernel functions and corresponding computational results are shown in Table 3.

From Table 3, several interesting results should be noted. First of all, the multiagent ensemble learning system dominated by the RBF-type kernel function produces the best classification results, followed by the sigmoid kernel and the polynomial kernel. Second, in terms of the total accuracy, there is a significant difference at the 10% level in the two-tail t-test between RBF kernel function and polynomial kernel function. Third, although the multiagent ensemble learning system dominated by the RBF kernel is advantageous to the ensemble learning system dominated by the sigmoid kernel, the robustness of the former is slightly worse than that of the latter (determined from the standard deviation measure). The possible reason for this is unknown, and worth exploring further.

4.2. Ensemble strategy testing

In this subsection, we mainly test the effects of ensemble strategy on the generalization performance of the multiagent ensemble learning system. As described in Section 3, bagging-based diversity strategy is used. Particularly, 1000 training datasets are replicated by the bagging algorithm in this section. In addition, the SVM models with RBF kernel function are used as intelligent learning agents. The parameters of SVM are determined by a grid search. Particularly, three main ensemble strategies: majority voting, *TA*-based weighted averaging and ALNN-based weighted averaging, are used here. The computational results are shown in Table 4.

From the results in Table 4, we draw the following conclusions.

(1) The ALNN-based adaptive weighting approach to ensemble learning produces a satisfactory result within the three ensemble strategies listed in this paper. This indicates that the ALNN-based adaptive weighting ensemble learning system is a promising ensemble strategy for use in a multiagent ensemble learning system for credit risk evaluation.
(2) Of the three ensemble strategies, the majority voting strategy is the worst. The main reason is that it ignores the existence of diversity that is the motivation for a multiagent ensemble learning system (Yang & Browne, 2004).
(3) Performance improvement ($71.19 - 68.62\% = 2.57\%$) of the ALNN-based adaptive weighting ensemble strategy relative to TA-based weighted averaging ensemble strategy is much better than that ($68.62 - 67.31\% = 1.31\%$) of the TA-based weighted averaging ensemble strategy relative to majority voting ensemble strategy.
(4) In terms of standard deviation, the TA-based weighted averaging ensemble strategy seems to be more stable than the ALNN-based ensemble strategies. The possible reason for

Table 3
Performance comparisons of SVM ensemble learning with kernel diversity.

No.	Kernel functions	Type I (%)	Type II (%)	Total accuracy (%)
1	600Poly+200Sig+200RBF	64.76 [4.76]	61.89 [5.05]	63.25 [4.94]
2	200Poly+600Sig+200 RBF	68.14 [4.28]	65.51 [4.69]	66.68 [4.55]
3	200Poly+200Sig+600 RBF	71.32 [4.93]	68.13 [4.71]	68.59 [4.87]

Table 4
Performance comparisons of SVM ensemble learning with different ensemble strategies.

No.	Ensemble strategies	Type I (%)	Type II (%)	Total accuracy (%)
1	Majority voting	68.63 [5.43]	66.24 [5.72]	67.31 [5.55]
2	TA-based weighted averaging	69.96 [4.65]	67.45 [4.21]	68.62 [4.37]
3	ALNN-based adaptive weighting	72.52 [4.63]	69.98 [4.44]	71.19 [4.53]

Table 5
Performance comparisons for different classification models.

System type	Model	Type I (%)	Type II (%)	Total accuracy (%)
Single agent system (SAS)	LDA	62.21 [5.22]	59.98 [4.86]	60.84 [5.00]
	QDA	64.18 [6.02]	63.12 [4.56]	64.32 [5.44]
	LogR	64.43 [5.14]	65.23 [5.67]	64.69 [5.46]
	FNN	60.43 [7.92]	57.54 [6.87]	58.85 [7.15]
	SVM	65.86 [5.21]	64.84 [4.54]	65.23 [4.88]
Multiagent system (MAS)	MV	68.63 [5.43]	66.24 [5.72]	67.31 [5.55]
	TA	69.96 [4.65]	67.45 [4.21]	68.62 [4.37]
	ALNN	72.52 [4.63]	69.98 [4.44]	71.19 [4.53]

Note: ALNN: ALNN-based adaptive weighting model; TA: TA-based weighted averaging model; MV: majority voting model; SVM: support vector machine model; LogR: logit regression model; QDA: quadratic discriminant analysis model; LDA: linear discriminant analysis model; FNN: feed-forward neural network model.

this is that the random initial weights of ALNN add some biases to the final output value. Further research is needed to confirm this.

4.3. Performance comparisons of different models

In terms of the previous settings, experiments to compare different classification models were conducted. Accordingly, the experimental results were obtained, and are shown in Table 5. Note that each class of experiment is repeated 1000 times. The final Type I, Type II and Total Accuracy figures show the average of the results of the 1000 individual tests and reflect model robustness. In addition, the values in brackets are standard deviations. Note that the parameters for single agent learning systems are determined by trial and error.

Based on the results in Table 5, several important conclusions can be drawn.

(1) Of the five single models, the SVM-based intelligent agent performs the best, followed by single logR, QDA, LDA, and FNN. The main reason for this is that the SVM-based intelligent agent utilizes a structural risk minimization or margin maximization principle, thus decreasing the risk of trapping into local minima, which often occurs in FNN training.
(2) In the multiagent learning systems, the proposed multiagent ALNN-based ensemble learning system consistently outperforms the other two similar multiagent learning models, implying that the proposed multiagent ALNN-based ensemble learning system is a promising approach to handle credit risk evaluation problem. Among the three multiagent learning models, the ALNN-based multiagent learning system performs the best, followed by TA-based weighted averaging model and majority voting based multiagent ensemble learning approach. There is no significant difference in performance of the three multiagent learning models. However, this may be due to some experimental difference. Confirmation of this requires further research.
(3) In the three listed multiagent learning models, all the multiagent ensemble learning systems perform better than the single agent models, i.e., SVM, logR, LDA, FNN and QDA. The main reason for this is that these multiagent learning systems used multiple SVM-based intelligent agents to obtain an ensemble result. Comparatively, single agent models such as FNN and SVM did not use any ensemble strategy.
(4) Comparing the three multiagent ensemble learning models with the best two single learning models, there are no significant differences at the 1% significance level. There are two possible reasons. One is the small size of the training samples. In our experiment, only 800 samples were used for training and the others were used for the purpose of validation and testing. Particularly, when we used two-fold cross validation to train the SVM agent, only 400 training data points were used. Such small-sized training data will weaken the generalization of the SVM agent. The other reason is the low diversity of the agents. Diversity is an important factor which affects the performance of the multiagent ensemble learning system. If all the agents in the multiagent ensemble learning system perform in the same way for each sample, the ensemble strategies will loss their value and lower the generalization performance.

From Table 5 and three measurements (i.e., Type I accuracy, II accuracy and total accuracy), we can judge which model is the best and which model is the worst. However, it is unclear what the differences between good models and bad ones are. For this purpose, McNemar's test (McNemar, 1947) is conducted to examine whether the proposed multiagent ensemble learning system significantly outperforms the other six models listed in this paper.

As a non-parametric test for two related samples, McNemar's test is particularly useful for before-after measurement of the same subjects (Cooper & Emory, 1995). Taking the total accuracy results from Tables 5 and 6 shows the results of the McNemar's test for credit dataset to statistically compare the performance in respect of testing data among the seven models. Note that the results listed in Table 6 are the Chi squared values and p values are in brackets.

Table 6
McNemar's test for pairwise performance comparison of different models.

Model	TA	WV	SVM	LogR	QDA	LDA	FNN
ALNN	0.9250 [0.3361]	2.1610 [0.1416]	4.8900 [0.0270]	5. 8230 [0.0158]	6.4170 [0.0113]	13.706 [0.0002]	18.831 [0.0001]
TA		0.6415 [0.9131]	1.4600 [0.2269]	1.9900 [0.1584]	2.3440 [0.1258]	7.3110 [0.0069]	11.182 [0.0008]
WV			0.4910 [0.4833]	0.8160 [0.3663]	1.0490 [0.3058]	4.8320 [0.0279]	8.0620 [0.0045]
SVM				0.0260 [0.8714]	0.0800 [0.7776]	2.1260 [0.1448]	4.4110 [0.0357]
LogR					0.0060 [0.9359]	1.5780 [0.2091]	3.6050 [0.0576]
QDA						1.2900 [0.2561]	3.1630 [0.0753]
LDA							0.3660 [0.5451]

Note: ALNN: ALNN-based adaptive weighting model; TA: TA-based weighted averaging model; MV: majority voting model; SVM: support vector machine model; LogR: logit regression model; QDA: quadratic discriminant analysis model; LDA: linear discriminant analysis model; FNN: feed-forward neural network model.

According to the results reported in Table 6, some important conclusions can be drawn in terms of McNemar's statistical test.

(1) The proposed ALNN-based adaptive weighting model outperforms the standard SVM, logit regression (LogR), quadratic discriminant analysis (QDA) model, linear discriminant analysis (LDA) model and feed-forward neural network (FNN) model at 1% statistical significance level. However, the proposed LS-FSVM model does not significantly outperform the TA-based weighted averaging model and majority voting based ensemble learning model. These results are consistent with those of Table 5.
(2) The TA-based weighted averaging model and weight voting based model outperforms LDA and FNN at 5% significance level. But the McNemar's test does not conclude that the TA-based weighted averaging model and weight voting model perform better than the other models.
(3) For SVM, LogR and QDA models, we can find that these three models perform much better than the FNN models at 10% significance level. In McNemar's statistical test, the FNN performs the poorest. The possible reason is that the FNN model suffers from the local minima or over-fitting problem.
(4) Comparing with LDA and FNN models, it is easy to find that the LDA model did not perform better than the FNN models at 10% significance level. All findings are consistent with results reported in Table 5.

In summary, according to the above experimental results and statistical testing, we can conclude that the ALNN-based adaptive weighting ensemble learning model can significantly outperform all models listed in this paper, revealing that the ALNN-based adaptive weighting ensemble learning model can also be used as a competitive solution to credit risk evaluation.

5. Conclusions

In this paper, a multi-stage SVM-based multiagent ensemble learning system was proposed for credit risk evaluation problems. For the purposes of verification, one publicly available credit dataset was used in order to test effectiveness and classification power. In particular, multiple different experiments were conducted to test the impact of various diversity strategies and ensemble strategies on the performance of a SVM-based ensemble learning system. All results reported in the experiments clearly show that the proposed SVM-based multiagent ensemble learning approach can consistently outperform the other comparable models, including the other two ensemble learning models and the five single agent learning systems. The obtained results reveal that the proposed SVM-based multiagent ensemble learning model can provide a promising solution to credit risk evaluation problem and implies that the proposed SVM-based multiagent ensemble learning technique has a great potential in its application to other classification problems.

Acknowledgements

This work is partially supported by grants from the National Natural Science Foundation of China (No. 70221001), the Knowledge Innovation Program of the Chinese Academy of Sciences, Grant-in-Aid for Science Research (No. 19500070) and MEXT.ORC (2004–2008) of Japan, and the NSFC/RGC Joint Research Scheme (No. N_CityU110/07).

References

Abdou, H., Pointon, J., & El-Masry, A. (2008). Neural nets versus conventional techniques in credit scoring in Egyptian banking. *Expert Systems with Applications, 35*(3), 1275–1292.
Bellotti, T., & Crook, J. (2009). Support vector machines for credit scoring and discovery of significant features. *Expert Systems with Applications, 36*(2), 3302–3308.
Breiman, L. (1996). Bagging predictors. *Machine Learning, 26*, 123–140.
Chen, M. C., & Huang, S. H. (2003). Credit scoring and rejected instances reassigning through evolutionary computation techniques. *Expert Systems with Applications, 24*(4), 433–441.
Chen, S. (2007). Computationally intelligent agents in economics and finance. *Information Sciences, 177*, 1153–1168.
Chen, W., Ma, C., & Ma, L. (2009). Mining the customer credit using hybrid support vector machine technique. *Expert Systems with Applications, 36*(4), 7611–7616.
Chen, W. H., & Shih, J. Y. (2006). A paper of Taiwan's issuer credit rating systems using support vector machines. *Expert Systems with Applications, 30*(3), 427–435.
Cooper, D. R., & Emory, C. W. (1995). *Business research methods*. Chicago: Irwin.
Dawson, C. W., & Wilby, R. (1998). An artificial neural network approach to rainfall-runoff modeling. *Hydrological Sciences Journal, 43*(1), 47–66.
Elliott, R. J., & Filinkov, A. (2008). A self tuning model for risk estimation. *Expert Systems with Applications, 34*(3), 1692–1697.
Grablowsky, B. J., & Talley, W. K. (1981). Probit and discriminant functions for classifying credit applicants: A comparison. *Journal of Economic Business, 33*, 254–261.
Hagan, M. T., Demuth, H. B., & Beale, M. H. (1996). *Neural network design*. Boston: PWS Publishing Company.
Henley, W. E., & Hand, D. J. (1996). A k-NN classifier for assessing consumer credit risk. *Statistician, 45*, 77–95.
Huang, Z., Chen, H. C., Hsu, C. J., Chen, W. H., & Wu, S. S. (2004). Credit rating analysis with support vector machines and neural networks: A market comparative paper. *Decision Support Systems, 37*, 543–558.
Kim, K. J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing, 55*, 307–319.
Kim, Y. S., & Sohn, S. Y. (2004). Managing loan customers using misclassification patterns of credit scoring model. *Expert Systems with Applications, 26*(4), 567–573.
Lai, K. K., Yu, L., Huang, W., & Wang, S. Y. (2006a). A novel support vector machine metamodel for business risk identification. *Lecture Notes in Artificial Intelligence, 4099*, 980–984.
Lai, K. K., Yu, L., Wang, S. Y., & Zhou, L. G. (2006b). Credit risk analysis using a reliability-based neural network ensemble model. *Lecture Notes in Computer Science, 4132*, 682–690.
Lai, K. K., Yu, L., Zhou, L. G., & Wang, S. Y. (2006c). Credit risk evaluation with least square support vector machine. *Lecture Notes in Artificial Intelligence, 4062*, 490–495.
Lai, K. K., Yu, L., Zhou, L. G., & Wang, S. Y. (2006d). Neural network metalearning for credit scoring. *Lecture Notes in Computer Science, 4113*, 403–408.
Lee, T. S., Chiu, C. C., Lu, C. J., & Chen, I. F. (2002). Credit scoring using the hybrid neural discriminant technique. *Expert Systems with Application, 23*(3), 245–254.
Lim, M. K., & Sohn, S. Y. (2007). Cluster-based dynamic scoring model. *Expert Systems with Applications, 32*(2), 427–431.
Lin, S. L. (2009). A new two-stage hybrid approach of credit risk in banking industry. *Expert Systems with Applications, 36*(4), 8333–8341.
Maier, H. R., & Dandy, G. C. (2000). Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications. *Environmental Modelling and Software, 15*, 101–124.
Malhotra, R., & Malhotra, D. K. (2003). Evaluating consumer loans using neural networks. *Omega, 31*, 83–96.
Martens, D., Baesens, B., Van Gestel, T., & Vanthienen, J. (2007). Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research, 183*(3), 1466–1476.
McNemar, Q. (1947). Note on the sampling error of differences between correlated proportions and percentages. *Psychometrica, 12*, 153–157.
Mues, C., Baesens, B., Files, C. M., & Vanthienen, J. (2004). Decision diagrams in machine learning: An empirical paper on real-life credit-risk data. *Expert Systems with Applications, 27*(2), 257–264.
Olmeda, I., & Fernandez, E. (1997). Hybrid classifiers for financial multicriteria decision making: The case of bankruptcy prediction. *Computational Economics, 10*, 317–335.
Quah, J. T. S., & Sriganesh, M. (2008). Real-time credit card fraud detection using computational intelligence. *Expert Systems with Applications, 35*(4), 1721–1732.
Partridge, D., & Yates, W. B. (1996). Engineering multiversion neural-net systems. *Neural Computation, 8*, 869–893.
Piramuthu, S. (1999). Financial credit-risk evaluation with neural and neurofuzzy systems. *European Journal of Operational Research, 112*, 310–321.
Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5*, 197–227.
Smalz, R., & Conrad, M. (1994). Combining evolution with credit apportionment: A new learning algorithm for neural nets. *Neural Networks, 7*, 341–351.
Tay, F. E. H., & Cao, L. J. (2001). Application of support vector machines in financial time series forecasting. *Omega, 29*, 309–317.
Thomas, L. C. (2002). A survey of credit and behavioral scoring: Forecasting financial risk of lending to consumers. *International Journal of Forecasting, 16*, 149–172.
Thomas, L. C., Edelman, D. B., & Crook, J. N. (2002). *Credit scoring and its applications*. Philadelphia: Society of Industrial and Applied Mathematics.
Thomas, L. C., Oliver, R. W., & Hand, D. J. (2005). A survey of the issues in consumer credit modelling research. *Journal of the Operational Research Society, 56*, 1006–1015.

Van Gestel, T., Baesens, B., Garcia, J., & Van Dijcke, P. (2003). A support vector machine approach to credit scoring. *Bank en Financiewezen, 2*, 73–82.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer.

Varetto, F. (1998). Genetic algorithms applications in the analysis of insolvency risk. *Journal of Banking and Finance, 22*, 1421–1439.

Wang, Y. Q., Wang, S. Y., & Lai, K. K. (2005). A new fuzzy support vector machine to evaluate credit risk. *IEEE Transactions on Fuzzy Systems, 13*, 820–831.

Wiginton, J. C. (1980). A note on the comparison of logit and discriminant models of consumer credit behaviour. *Journal of Financial Quantitative Analysis, 15*, 757–770.

Yang, S., & Browne, A. (2004). Neural network ensembles: Combining multiple models for enhanced performance using a multistage approach. *Expert Systems, 21*, 279–288.

Yu, L., Lai, K. K., Wang, S. Y., & Huang, W. (2006). A bias-variance-complexity trade-off framework for complex system modeling. *Lecture Notes in Computer Science, 3980*, 518–527.

Yu, L., Wang, S. Y., & Lai, K. K. (2005). A novel nonlinear ensemble forecasting model incorporating GLAR and ANN for foreign exchange rates. *Computers and Operations Research, 32*(10), 2523–2541.

Yu, L., Wang, S. Y., & Lai, K. K. (2007). An intelligent-agent-based fuzzy group decision making model for financial multicriteria decision support: The case of credit scoring. *European Journal of Operational Research, 195*(3), 942–959.

Yu, L., Wang, S. Y., & Lai, K. K. (2008). Credit risk assessment with a multistage neural network ensemble learning approach. *Expert Systems with Applications, 34*(2), 1434–1444.

Yu, L., Wang, S. Y., Lai, K. K., & Zhou, L. G. (2008). *Bio-inspired credit risk analysis – computational intelligence with support vector machines*. Berlin: Springer-Verlag.