Knowledge-Based Systems 24 (2011) 1348–1359

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A novel bankruptcy prediction model based on an adaptive fuzzy *k*-nearest neighbor method

Hui-Ling Chen, Bo Yang, Gang Wang, Jie Liu, Xin Xu, Su-Jing Wang, Da-You Liu *

College of Computer Science and Technology, Jilin University, Changchun 130012, China
Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China

ARTICLE INFO

Article history:
Received 11 October 2010
Received in revised form 3 June 2011
Accepted 9 June 2011
Available online 25 June 2011

Keywords:
Fuzzy *k*-nearest neighbor
Parallel computing
Particle swarm optimization
Feature selection
Bankruptcy prediction

ABSTRACT

Bankruptcy prediction is one of the most important issues in financial decision-making. Constructing effective corporate bankruptcy prediction models in time is essential to make companies or banks prevent bankruptcy. This study proposes a novel bankruptcy prediction model based on an adaptive fuzzy *k*-nearest neighbor (FKNN) method, where the neighborhood size *k* and the fuzzy strength parameter *m* are adaptively specified by the continuous particle swarm optimization (PSO) approach. In addition to performing the parameter optimization for FKNN, PSO is also utilized to choose the most discriminative subset of features for prediction. Adaptive control parameters including time-varying acceleration coefficients (TVAC) and time-varying inertia weight (TVIW) are employed to efficiently control the local and global search ability of PSO algorithm. Moreover, both the continuous and binary PSO are implemented in parallel on a multi-core platform. The proposed bankruptcy prediction model, named PTVPSO-FKNN, is compared with five other state-of-the-art classifiers on two real-life cases. The obtained results clearly confirm the superiority of the proposed model in terms of classification accuracy, Type I error, Type II error and area under the receiver operating characteristic curve (AUC) criterion. The proposed model also demonstrates its ability to identify the most discriminative financial ratios. Additionally, the proposed model has reduced a large amount of computational time owing to its parallel implementation. Promisingly, PTVPSO-FKNN might serve as a new candidate of powerful early warning systems for bankruptcy prediction with excellent performance.

1. Introduction

Accurately identifying the potentially financial failure of companies remains a goal of many stakeholders involved. Because there is no underlying economic theory of bankruptcy, searching for more accurate bankruptcy prediction models remains the goal in the field of the bankruptcy prediction. As a matter of fact, bankruptcy prediction can be formulated as the problem of solving classification task. A fair amount of classification models has been developed for bankruptcy prediction. These models have progressed from statistical methods to the artificial intelligence (AI) approaches. A number of statistical methods such as the simple univariate analysis [1], multivariate discriminant analysis technique [2], logistic regression approach [3] and factor analysis technique [4] have been typically used for financial applications including bankruptcy prediction. Recent studies in the AI approach, such as artificial neural networks (ANN) [5–11], rough set theory [12–14], support vector machines (SVM) [15–17], *k*-nearest neighbor method (KNN) [18–20], Bayesian network models [21,22] and other different methods such as hybrid methods and ensemble methods [23–26] have also been successfully applied to bankruptcy prediction (see [25,26] for detail). Among these techniques, ANN has become one of the most popular techniques for the prediction of corporate bankruptcy due to its high prediction accuracy. However, a major disadvantage of ANN lies in their knowledge representation. The black box nature of ANN makes it difficult for humans to understand how the networks predict the bankruptcy.

Compared to ANN, KNN is simple, easily interpretable and can achieve acceptable accuracy rate. Albeit these advantages, the standard KNN methods place equal weights on all the selected neighbors regardless of their distances from the query point. An improvement over the standard KNN classifier is the fuzzy *k*-nearest neighbor classifier (FKNN) [27], which uses concepts from fuzzy logic to assign degree of membership to different classes while considering the distance of its *k*-nearest neighbors. It means that all the instances are assigned a membership value in each class rather than binary decision of 'bankruptcy' or 'non-bankruptcy'.

* Corresponding author at: Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China.
E-mail addresses: liudy@jlu.edu.cn, liudayou19420601@gmail.com (D.-Y. Liu).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.06.008

Points closer to the query point contributes larger value to be assigned to the membership function of their corresponding class in comparison to far away neighbors. The class with the highest membership function value is taken as the winner. The FKNN method has been frequently used for the classification of biological data [28–30], image data [31,32] and so on. Nevertheless, only few works have paid attention to using FKNN to deal with the financial problems. Bian and Mazlack [33] used FKNN as a reference classifier in their experiments in order to show the superiority of the proposed fuzzy-rough KNN method, which incorporated the rough set theory into FKNN to further improve the accuracy of bankruptcy prediction. However, they did not comprehensively investigate the neighborhood size k and the fuzzy strength parameter m, which play a significant role in improving the prediction result. This work will explore the full potential of FKNN by automatically determining k and m to exploit the maximum classification accuracy for bankruptcy prediction.

Besides choosing a good learning algorithm, feature selection is also an important issue in building the bankruptcy prediction models [25,34–37], which refers to choosing subset of attributes from the set of original attributes. The purpose of the feature selection is to identify the significant features and build a good learning model. The benefits of feature selection are threefold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data [38]. In bankruptcy prediction, genetic algorithms (GA) are usually used to select a subset of input features [39–41], to find appropriate hyper-parameter values of a predictor (for example, the kernel width and the regularization constant in the case of SVM) [35,42,43], or to determine predictor parameters (for example, multilayer perceptron weights) [44,45]. Compared with GA, PSO algorithm [46] has no crossover and mutation operators, it is simple and computationally inexpensive both in memory and runtime. Additionally, every particle adjusts their velocity and position according to the local best and global best. So that all the particles have a powerful search capability, which can help the swarm find the optimal solution. As for GA, after finding a locally optimum, it is difficult for it to find out a much better one even with a random search strategy in terms of mutation operator especially within a reasonable searching time. In this work, we will focus on exploring the PSO-based parameter optimization and feature selection approach. The continuous PSO algorithm will be employed to evolve an adaptive FKNN, where the neighborhood size k and the fuzzy strength parameter m are adaptively specified. On the other hand, the binary PSO will be used as a feature selection vehicle to identify the most informative features as well.

When dealing with the practical problems, the evolutionary-based methods such as the PSO and GA will cost a lot of computational time. There is an urgent need to improve the performance using high-performance computing techniques. For this reason, it is one of the major purposes of this paper to use a parallel environment to speed up the search and optimization process. Both the continuous and binary PSO are implemented on a multi-core platform using OpenMP (Open Multi-Processing) which is a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms [47]. The efficiency and effectiveness of the proposed bankruptcy prediction model is validated by comparing with other five state-of-the-art classification methods on two real-life cases. The experimental results demonstrate that the proposed model can not only obtain the most appropriate parameters but also show high discriminating power as a feature selection tool. Further comparison is also made between the parallel model and serial one. Based on the experiments conducted, it is inferred that the parallel model PTVPSO-FKNN can significantly reduce the computational time.

The rest of the paper is organized as follows. In Section 2, we give a brief description of the FKNN method and PSO algorithm. Section 3 proposes our model, the simultaneous optimization of relevant parameters and feature subset by the PSO approach in a parallel environment. In the next section, the detailed experimental design is presented, and Section 5 describes all the empirical results and discussion. Finally, conclusions and future work are summarized in Section 6.

2. Background materials

2.1. Fuzzy k-nearest neighbor algorithm

The k-nearest neighbor algorithm (KNN) is one of the oldest and simplest non-parametric pattern classification methods. In the KNN algorithm a class is assigned according to the most common class amongst its k-nearest neighbors. In 1985, Keller proposed a fuzzy version of KNN by incorporating the fuzzy set theory into the KNN algorithm, and named it as "fuzzy KNN classifier algorithm" (FKNN) [27]. According to his approach, rather than individual classes as in KNN, the fuzzy memberships of samples are assigned to different categories according to the following formulation:

$$u_i(x) = \frac{\sum_{j=1}^{k} u_{ij}\left(1/\|x - x_j\|^{2/(m-1)}\right)}{\sum_{j=1}^{k}\left(1/\|x - x_j\|^{2/(m-1)}\right)} \tag{1}$$

where $i = 1, 2, \ldots C$, and $j = 1, 2, \ldots, k$, with C number of classes and k number of nearest neighbors. The fuzzy strength parameter m is used to determine how heavily the distance is weighted when calculating each neighbor's contribution to the membership value, and its value is usually chosen as $m \in (1, \infty)$. $\|x - x_j\|$ is the distance between x and its jth nearest neighbor x_j. Various metrics can be chosen for $\|x - x_j\|$, such as Euclidean distance, Hamming distance, and Mahalanobis distance, among other distances. In this study, the Euclidean metric is used. u_{ij} is the membership degree of the pattern x_j from the training set to the class i, among the k nearest neighbors of x. There are two ways [27] to define u_{ij}, one way is the crisp membership, i.e., each training pattern has complete membership in their known class and non-memberships in all other classes. The other way is the constrained fuzzy membership, i.e., the k nearest neighbors of each training pattern (say x_k) are found, and the membership of x_k in each class is assigned as:

$$u_{ij}(x_k) = \begin{cases} 0.51 + (n_j/K)^{*}0.49, & \text{if } j = i \\ (n_j/K)^{*}0.49, & \text{if } j \neq i \end{cases} \tag{2}$$

The value n_j is the number of neighbors found which belong to the jth class. Note that, the memberships calculated by Eq. (2) should satisfy the following equations:

$$\sum_{i=1}^{C} \mu_{ij} = 1, \quad j = 1, 2, \ldots, n \tag{3}$$

$$0 < \sum_{j=1}^{n} u_{ij} < n \tag{4}$$

$$u_{ij} \in [0, 1] \tag{5}$$

In our experiments, we have found that the second way leads to better classification accuracy. After calculating all the memberships for a query sample, it is assigned to the class with which it has the highest membership value, i.e.,

$$C(x) = \arg\max_{i=1}^{C}(u_i(x)) \tag{6}$$

The pseudo-code of the FKNN algorithm is given below:

Input: (a) The training set X with the labeled patterns $\{x_i | i = 1,2,\ldots,n\}$.
(b) The test pattern y.
Output: (a) Class label of y.
(b) Confidence for each class label.
ALGORITHM:
For $i = 1, 2, \ldots,$ to n
Compute the distance from x_i to y using the Euclidean metric.
If $i \leqslant k$
Include xi in the set of k nearest neighbors.
Else if (x_i is closer to y than any previous nearest neighbors)
Delete the farthest of the k nearest neighbors.
Include x_i in the set of k nearest neighbors.
End if
End for
For $c = 1$ to C
Compute $u_i(x)$ using (1).
End for
Crisp class label of y is assigned to the class with which it has the highest membership value using (6).

2.2. Time variant particle swarm optimization (TVPSO)

PSO is inspired by the social behavior of organisms such as bird flocking and fish schooling, which was first developed by Kennedy and Eberhart [46,48]. In PSO each individual is treated as a particle in d-dimensional space, and each particle has a position and velocity. The position vector of the ith particle is represented as $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$, and its according velocity is represented as $V_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,d})$. The velocity and position are updated as follows:

$$v_{i,j}^{n+1} = w \times v_{i,j}^{n} + c_1 \times r_1 \left(p_{i,j}^{n} - x_{i,j}^{n}\right) + c_2 \times r_2 \left(p_{g,j}^{n} - x_{i,j}^{n}\right) \quad (7)$$

$$x_{i,j}^{n+1} = x_{i,j}^{n} + v_{i,j}^{n+1}, \quad j = 1,2,\ldots,d \quad (8)$$

where vector $P_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,d})$ represents the best previous position of the ith particle that gives the best fitness value, which is known as the personal best position (*pbest*). Vector $P_g = (p_{g,1}, p_{g,2}, \ldots, p_{g,d})$ is the best particle among all the particles in the population, which is known as the global best position (*gbest*). r_1 and r_2 are random numbers, generated uniformly in the range $[0,1]$. The velocity $v_{i,j}$ is restricted to the range $[-v_{max}, v_{max}]$, in order to prevent the particles from flying out of the solution space. Generally, $v_{\max}$ is suggested to set to be 10–20% of the dynamic range of the variable in each dimension [49].

Inertia weight w, introduced by Shi and Eberhart, which is used to balance the global exploration and local exploitation [50]. A large inertia weight facilitates the global search, while a small inertia weight facilitates the local search. In order to reduce the weight over the iterations allowing the algorithm to exploit some specific areas, the inertia weight w is updated according to the following equation:

$$w = w_{\min} + (w_{\max} - w_{\min}) \frac{(t_{\max} - t)}{t_{\max}} \quad (9)$$

where $w_{\max}$, $w_{\min}$ are the predefined maximum and minimum values of the inertia weight w, t is the current iteration of the algorithm and $t_{\max}$ is the maximum number of iterations. Usually the value of w is varied between 0.9 and 0.4. Eq. (9) is also known as time-varying inertia weight (TVIW), which will be incorporated into the TVPSO. It has been shown to significantly improve the performance of PSO [51], since it makes PSO have more global search ability at the beginning of the run and have more local search ability near the end of the run. c_1 and c_2 are acceleration coefficients, which define the magnitude of the influences on the particles velocity in the directions of the personal and the global optima, respectively. To better balance the search space between the global exploration and local exploitation, time-varying acceleration coefficients (TVAC) have been introduced in [52]. This concept will be adopted in this study to ensure the better search for the solutions. The core idea of TVAC is that decreases from its initial value of c_{1i} to c_{1f}, while increases from c_{2i} to c_{2f} using the following equations as in [52]. TVAC can be mathematically represented as follows:

$$c_1 = (c_{1f} - c_{1i}) \frac{t}{t_{\max}} + c_{1i}, \quad (10)$$

$$c_2 = (c_{2f} - c_{2i}) \frac{t}{t_{\max}} + c_{2i}. \quad (11)$$

where, c_{1f}, c_{1i}, c_{2f}, and c_{2i} are constants, t is the current iteration of the algorithm and $t_{\max}$ is the maximum number of iterations.

For the binary PSO, the discrete PSO version introduced by Kennedy and Eberhart [53] was adopted in this study. The binary PSO is searching in a discrete space (i.e., searching in a space where '0' presents the feature is selected '1' denotes the feature is discarded). Where a particle moves in a state space restricted to zero and one on each dimension, in terms of the changes in probabilities that a bit will be in one state or the other. If the velocity is high it is more likely to choose '1', and lower values favor choosing '0'. A sigmoid function is applied to transform the velocity from continuous space to probability space:

$$sig(v_{i,j}) = \frac{1}{1 + \exp(-v_{i,j})}, \quad j = 1,2,\ldots,d \quad (12)$$

The velocity update Eq. (7) stays unchanged except that $x_{i,j}$, $p_{i,j}$ and $p_{g,j} \in \{0,1\}$, and in order to ensure that bit can transfer between '1' and '0' with a positive probability, $v_{\max}$ was introduced to limit $v_{i,j}$. The new particle position is updated using the following rule:

$$x_{i,j}^{n+1} = \begin{cases} 1, & \text{if } rnd < sig(v_{i,j}) \\ 0, & \text{if } rnd \geqslant sig(v_{i,j}) \end{cases}, \quad j = 1,2,\ldots,d \quad (13)$$

where $sig(v_{i,j})$ is calculated according to Eq. (12), rnd is a uniform random number in the range $[0,1]$.

As described above, TVPSO is adaptive in nature by allowing its inertia weight and acceleration coefficients to vary with iterations during its search in the continuous and discrete space. This character helps the algorithm explore the search space to a greater extent.

3. Proposed PTVPSO-FKNN prediction model

In this section, we describe the proposed PTVPSO-FKNN model for bankruptcy prediction. As mentioned in the Introduction, the aim of this model is to optimize the FKNN classifier by automatically: (1) determining the number of nearest neighbors k and the fuzzy strength parameter m and (2) identifying the subset of best discriminative features. In order to achieve this goal, the continuous and binary PSO are combined together to dynamically conduct parameter optimization and feature selection simultaneously. The obtained appropriate feature subset is served as the input into the optimized FKNN model for classification. PTVPSO-FKNN takes into consideration two fitness values for parameter optimization and feature selection. One is the AUC value and the other is the number of selected features by TVPSO. Here, we first describe the model based on the serial PSO algorithm, termed TVPSO-FKNN, and then implement it in parallel.

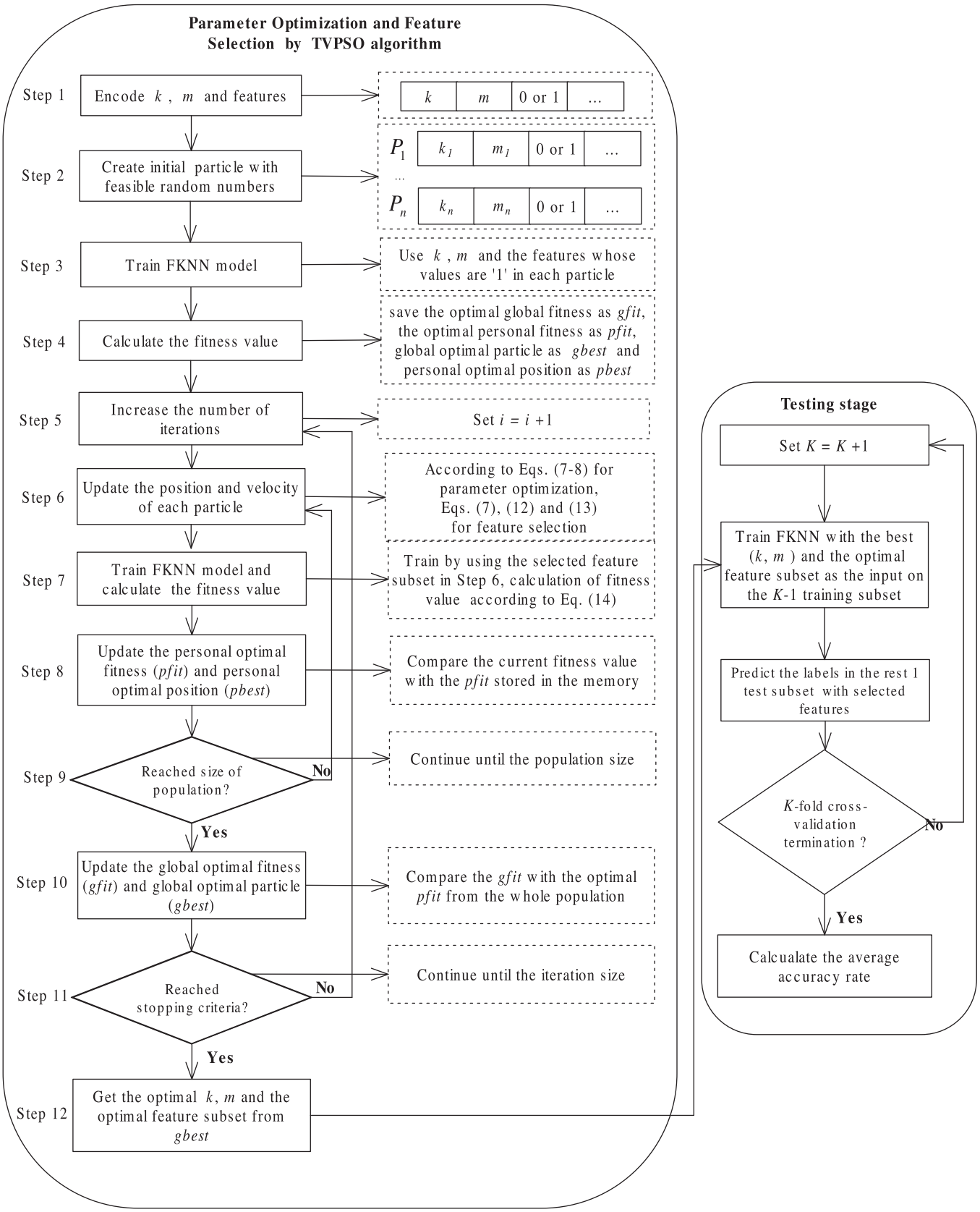

Fig. 1. Flowchart of the TVPSO-FKNN model.

3.1. TVPSO-FKNN model based on the serial PSO algorithm

The flowchart of the TVPSO-FKNN model for bankruptcy prediction was constructed through the following main steps as shown in Fig. 1.

Step 1: Encode the particle with $n + 2$ dimensions. The first two dimensions are k and m which are continuous values. The remaining n dimensions is Boolean features mask, which is represented by discrete value, '1' indicates the feature is selected, and '0' represents the feature is discarded.

Step 2: Initialize the individuals of the population with random numbers. Meanwhile, specify the PSO parameters including the lower and upper bounds of the velocity, the size of particles, the number of iterations, etc.

Step 3: Train the FKNN model with the selected features.

Step 4: It is well known that higher the AUC value the better the classifier is said to be. The particle with high AUC value and the small number of selected features can produce a high fitness value. Hence, we took both of them into consideration in designing the objective function, the fitness value was calculated according to the following objective function:

$$\begin{cases} f_1 = \text{AUC} \\ f_2 = \left(1 - \frac{\sum_{j=1}^{n} ft_i}{n}\right) \\ f_{=} \alpha \times f_1 + \beta \times f_2 \end{cases} \tag{14}$$

where variable AUC in the first sub-objective function f_1 represents the area under the ROC curve achieved by the FKNN classifier via K-fold cross-validation (CV), here $K = 5$. Note that here the 5-fold CV is used to determine the optimal parameters (including k and m) which is different from the outer loop of 10-fold CV, which is used to do the performance estimation. In the second sub-objective function f_2, ft_i is the value of feature mask ('1' represents that feature is selected and '0' indicates that feature is discarded), n is the total number of features. The weighted summation of the two sub-objective functions is selected as the final objective function. In the function f, α is the weight for FKNN classification accuracy, β indicates the weight for the selected features. The weight can be adjusted to a proper value depends on the importance of the sub-objective function. Because the classification performance depends more on the classification accuracy, hence the α value is set as much bigger than that of β. According to our preliminary experiments, the values of α and β were taken as 0.85 and 0.15, respectively. After the fitness value was obtained, the global optimal fitness was saved as *gfit*, personal optimal fitness as *pfit*, global optimal particle as *gbest* and personal optimal particle as *pbest*.

Step 5: Increase the number of iteration.

Step 6: Increase the number of population. Update the position and velocity of k, m using Eqs. (7) and (8) and the features using Eqs. (7), (12) and (13) in each particle.

Step 7: Train the FKNN classifier with the feature vector obtained in Step 6 and calculate the fitness value of each particle according to Eq. (14). Notice that PSO is used for optimization tasks where the neighborhood size k to be optimized is integer number. Hence, an extra step is taken to round the encoded value k to the nearest integer number before the particle is evaluated.

Step 8: Update the personal optimal fitness (*pfit*) and personal optimal position (*pbest*) by comparing the current fitness value with the *pfit* stored in the memory. If the current fitness is dominated by the *pfit* stored in the memory, then keep the *pfit* and *pbest* in the memory; otherwise, replace the *pfit* and *pbest* in the memory with the current fitness value and particle position.

Step 9: If the size of the population is reached, then go to Step 10. Otherwise, go to Step 6.

Step 10: Update the global optimal fitness (*gfit*) and global optimal particle (*gbest*) by comparing the *gfit* with the optimal *pfit* from the whole population, If the current optimal *pfit* is dominated by the *gfit* stored in the memory, then keep the *gfit* and *gbest* in the memory; otherwise, replace the *gfit* and *gbest* in the memory with the current optimal *pfit* and the optimal *pbest* from the whole population.

Step 11: If the stopping criteria are satisfied, then go to Step 12. Otherwise, go to Step 5. The termination criteria are that the iteration number reaches the maximum number of iterations or the value of *gfit* does not improve after 100 consecutive iterations.

Step 12: Get the optimal (k,m) and feature subset from the best particle (*gbest*).

3.2. Parallel implementation of the TVPSO-FKNN (PTVPSO-FKNN)

When dealing with the practical problems, the evolutionary-based methods such as PSO and GA will cost a lot of computational time. There is an urgent need to improve the performance using high-performance computing techniques. Consequently, we attempt to implement TVPSO-FKNN in parallel on multi-core processor by using OpenMP to speed up the search and optimization process.

The architecture of the multi-core platform is divided into three lays as shown in Fig. 2: (1) TVPSO-FKNN: It consists of a number of particles, which can supply computing requirements. The parallel algorithm controls the iterations of particles and each particle is calculated separately. (2) OpenMP: This component guarantees to implement parallel synchronization and establish the communications with operating system (OS). The main part of OpenMP is scheduler, which provides the system with job scheduling and allocation. (3) Multi-core processor: The job is dispatched by OpenMP via OS.

The pseudo-code of the parallel PTVPSO-FKNN is as follows:

```
Initialize system parameters
Train FKNN model
Calculate fitness
While (cni < mni)/*current number of iteration (cni),
   maximum number of iteration (mni).*/
  For each particle
      Update position
      Update velocity
      Train FKNN model
      Calculate fitness
      Calculate pfit. /* personal optimal fitness (pfit)*/
      Calculate pbest. /* personal optimal position (pbest)*/
  End for
      Calculate gfit./*global optimal fitness (gfit)*/
      Calculate gbest./*global optimal particle (gbest)*/
  cni = cni + 1.
End while
```

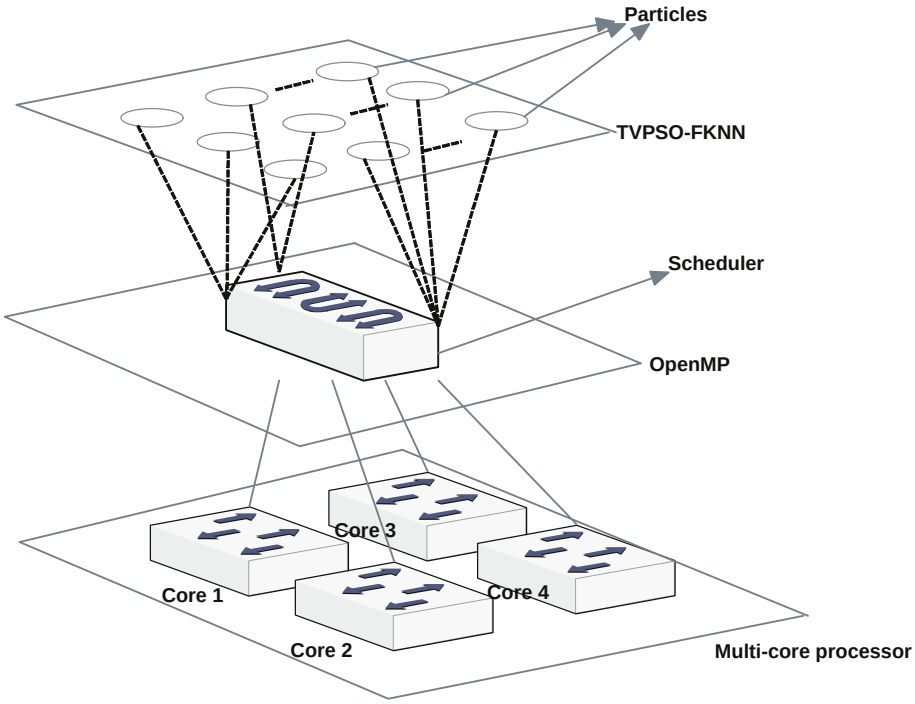

Fig. 2. Parallel architecture of the PTVPSO-FKNN model.

4. Experimental design

4.1. Data description

The first financial data used for this study is the Wieslaw [54] dataset which contains 30 financial ratios and 240 cases in total (112 from bankrupt Polish companies and 128 from non-bankrupt ones between 1997 and 2001). All the observations cover the period spanning 2 to 5 yr before bankruptcy took place. It should be noted that the size of the dataset is not that large compared to the majority of bankruptcy prediction studies. However, according to [55], the dataset is reliable since increasing the dataset length does not lead to the accuracy increase. The description of the 30 financial ratios is shown in Table 1. Fig. 3 illustrates the distribution of the two classes of 240 samples in the subspace formed by the two best features according to the principal component analysis algorithm [56]. As shown in this figure, there is apparently strong overlap between the bankrupt companies and non-bankrupt ones.

Table 1
The description of the Wieslaw dataset.

No.	Predictor variable name	Financial ratios
X_1	Cash/current liabilities	C/CL
X_2	Cash/total assets	C/TA
X_3	Current assets/current liabilities	CA/CL
X_4	Current assets/total assets	CA/TA
X_5	Working capital/total assets	WC/TA
X_6	Working capital/sales	WC/S
X_7	Sales/inventory	S/I
X_8	Sales/receivables	S/R1
X_9	Net profit/total assets	NP/TA
X_{10}	Net profit/current assets	NP/CA
X_{11}	Net profit/sales	NP/S1
X_{12}	Gross profit/sales	GP/S
X_{13}	Net profit/liabilities	NP/L
X_{14}	Net profit/equity	NP/E
X_{15}	Net profit/(equity + long term liabilities)	NP/EL
X_{16}	Sales/receivables	S/R2
X_{17}	Sales/total assets	S/TA1
X_{18}	Sales/current assets	S/CA
X_{19}	(365*receivables)/sales	R/S
X_{20}	Sales/total assets	S/TA2
X_{21}	Liabilities/total income	L/TI
X_{22}	Current liabilities/total income	CL/TI
X_{23}	Receivables/liabilities	R/L
X_{24}	Net profit/sales	NP/S2
X_{25}	Liabilities/total assets	L/TA
X_{26}	Liabilities/equity	L/E
X_{27}	Long term liabilities/equity	LTL/E
X_{28}	Current liabilities/equity	CL/E
X_{29}	EBIT (earnings before interests and taxes)/total assets	EBIT/TA
X_{30}	Current assets/sales	CA/S

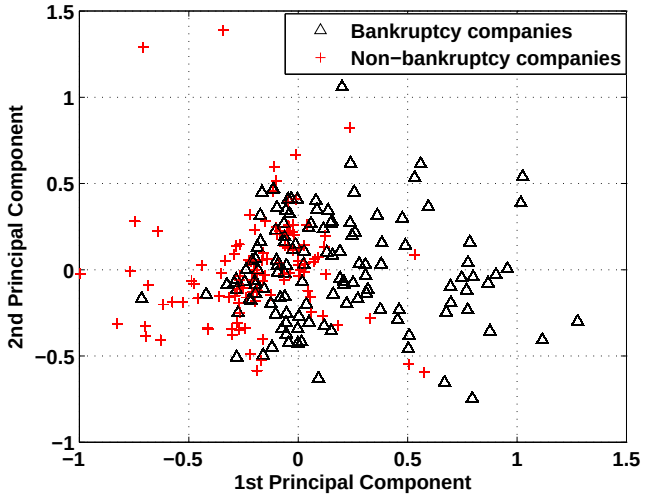

Fig. 3. Two-dimensional distribution of the two classes (bankrupt and non-bankrupt) in the subspace formed by the best couple of features obtained with the PCA algorithm.

The second dataset is the Australian credit dataset, which is available from the UCI Repository of Machine Learning Databases. The Australian credit data consists of 307 instances of creditworthy applicants and 383 instances where credit is not creditworthy. Each instance contains 6 nominal, 8 numeric attributes, and 1 class attribute (accepted or rejected). This dataset is interesting because there is a good mixture of attributes: continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values. To protect the confidentiality of data, the attributes names and values have been changed to meaningless symbolic data.

Normalization is employed to avoid feature values in greater numerical ranges dominating those in smaller numerical ranges, as well as to avoid the numerical difficulties during the calculation [57]. Generally, the data could be normalized by scaling them into the interval of [0,1] or [−1,1], here we chose the range of [−1,1] according to the Eq. (15), where x is the original value, x' is the scaled value, max_a is the maximum value of feature a, and min_a is the minimum value of feature a.

$$x' = \left(\frac{x - min_a}{max_a - min_a}\right)^{*} 2 - 1 \tag{15}$$

In order to gain an unbiased estimate of the generalization accuracy, the k-fold CV presented by Salzberg [58] was used to evaluate the classification accuracy. This study set k as 10, i.e., the data was divided into ten subsets. Each time, one of the 10 subsets is used as the test set and the other 9 subsets are put together to form a training set. Then the average error across all 10 trials is computed. The advantage of this method is that all of the test sets are independent and the reliability of the results could be improved. And we attempted to design our experiment using two loops. The inner loop is used to determine the optimal parameters and best feature subset for the FKNN classifier. The outer loop is used for estimating the performance of the FKNN classifier. In order to keep the same proportion of bankrupt and non-bankrupt companies of each set as that of the entire dataset, here a stratified 10-fold CV is employed as the outer loop and a stratified 9-fold CV is used for the inner loop. It is also referred to as the nested stratified 10-fold CV, which is also used in [59] for the microarray gene data analysis.

4.2. Experimental scheme

The proposed experimental framework was articulated around the following three main experiments.

(1) The first experiment aimed at assessing the effectiveness of the FKNN approach in bankruptcy prediction in the whole original feature space. For comparison purpose, we implemented five other reference classification approaches, namely KNN, SVM [60], back propagation neural network (BPNN), the probabilistic neural network (PNN) and extreme learning machine (ELM) [61]. In addition, we have implemented GA based FKNN (GA-FKNN) for comparison purpose.
(2) In the second experiment, it was plan to assess the capability of the proposed PTVPSO-FKNN model with feature selection to boost further the performance of the FKNN classifier by using the time-varying PSO approach. Furthermore, we attempted to investigate the whole evolutionary process of TVPSO in performing the parameter optimization and feature selection.
(3) The third experimental part had for objective to assess the capability of the proposed parallel TVPSO-FKNN model to enhance further the efficiency of the serial TVPSO-FKNN model with respect to the CPU time.

4.3. Experimental setup

The proposed PTVPSO-FKNN model is implemented using Microsoft Visual C++ 6.0 and OpenMP. For SVM, LIBSVM implementation is utilized, which is originally developed by Chang and Lin [62]. Regarding ELM, the implementation by Zhu and Huang available from http://www3.ntu.edu.sg/home/egbhuang is used. We implement PSO, GA, FKNN and KNN from scratch. BPNN and PNN are developed by using the Neural Network Toolbox of Matlab 7.0. The computer is Intel Quad-Core Xeon 2.0 GHz CPU; 4 GB RAM and the system is Windows Server 2003.

The detail parameter setting for PTVPSO-FKNN is as follows. The number of the iterations and particles are set to 250 and 8 for the Wieslaw dataset, 200 and 5 for the Australian dataset, respectively. The searching ranges for k and m are as follows: $k \in [1,100]$ and $m \in [1,10]$ for the Wieslaw dataset, $k \in [1,100]$ and $m \in [1,100]$ for the Australian dataset. v_{max} is set about 60% of the dynamic range of the variable on each dimension for the continuous type of dimensions. For the discrete type particle for feature selection, $[-v_{max}, v_{max}]$ is set as $[-6,6]$. As suggested in [52], c_{1i}, c_{1f}, c_{2i} and c_{2f} are set as follows: $c_{1i} = 2.5$, $c_{1f} = 0.5$, $c_{2i} = 0.5$, $c_{2f} = 2.5$. According to our preliminary experiment, w_{max} and w_{min} are set to 0.9 and 0.4, respectively.

For GA, the solution is binary-encoded and the roulette wheel selection algorithm is used. The crossover probability and mutation probability are set to 0.8 and 0.05, respectively. To perform a fair comparison, the same computational effort is used in TVPSO and GA. That is, the maximum generation, population size and searching range of the parameters in GA are the same as those in TVPSO. For SVM, we consider the nonlinear SVM based on the popular Gaussian (RBF) kernel, and a grid-search technique [57] is employed using 10-fold CV to find out the optimal parameter values of RBF kernel function. The range of the related parameters C and γ are varied between $C = \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\gamma = \{2^{-15}, 2^{-13}, \ldots, 2^{1}\}$. There will be $11 \times 9 = 99$ parameter combinations of (C,γ) are tried and the one with the best CV accuracy is chosen as the parameter values of the RBF kernel. Then the best parameter pair (C,γ) is used to create the model for training. For KNN, we find the best value of k within the range [1,100] by using 10-fold CV. Concerning BPNN, we use the three layer back-propagation network. We try different settings of the number of nodes in the hidden layers (5,10,15,20,25 and 30) and the different learning epochs (50,100,200 and 300) as the stopping criteria for training. In PNN, the pattern layer uses RBF neuron with spread parameter of 0.1 and 0.8 give the best accuracies by using the 10-fold CV on the Wieslaw dataset and Australian dataset, respectively. Hence these two values will be used for the subsequent analysis. In ELM the sigmoid activation function is used to compute the hidden layer output matrix. ELM models are built for 100 different numbers of neurons between 1 and 100. The best number of neurons will be taken to create the training model.

4.4. Measure for performance evaluation

Type I error, Type II error, total classification accuracy (ACC) and the area under the receiver operating characteristic curve (AUC) [63] were used to test the performance of the proposed PTVPSO-FKNN model. They are the most widely used measures to assess the performance of bankruptcy prediction systems [25]. Before defining these measures, we introduced the concept of confusion matrix, which is presented in Table 2. Where TP is the number of true positives, which means that some cases with 'positive' class (with bankruptcy) is correctly classified as positive; FN, the number of false negatives, which means that some cases with the 'positive' class is classified as negative; TN, the number of true

Table 2
Confusion matrix for bankruptcy prediction.

	Actual positive (Bankruptcy)	Actual negative (Non-Bankruptcy)
Predicted positive (Bankruptcy)	True positive (TP)	False positive (FP)
Predicted negative (Non-Bankruptcy)	False negative (FN)	True negative (TN)

negatives, which means that some cases with the 'negative' class (with non-bankruptcy) is correctly classified as negative; and FP, the number of false positives, which means that some cases with the 'negative' class is classified as positive.

Type I and Type II errors are two important measures which describe how well the classifier discriminates between case with non-bankruptcy and with bankruptcy. Type I error measures the proportion of non-bankrupt cases which are incorrectly identified as bankrupt ones. It is defined as Type I error = FP/(FP + TN). Type II error measures the proportion of bankrupt cases which are incorrectly identified as non-bankrupt ones. It is defined as Type II error = FN/(TP + FN). The ACC is calculated by TP + TN/(TP + FP + FN + TN). The receiver operating characteristic (ROC) curve is a graphical display that gives the measure of the predictive accuracy of a logistic model. The curve displays the true positive rate and false positive rate. AUC is the area under the ROC curve, which is one of the best methods for comparing classifiers in two-class problems.

5. Experimental results and discussion

5.1. Experiment I: classification in the whole original feature space

As mentioned earlier, in this experiment we evaluated the effectiveness of the proposed model on the original feature space. In order to verify the effectiveness of the proposed model, TVPSO-FKNN was compared with five other reference classifiers (SVM, KNN, BPNN, PNN and ELM). Tables 3 and 4 show the results achieved with all six investigated classifiers (PTVPSO-FKNN, SVM, KNN, BPNN, PNN and ELM) for the Wieslaw dataset and the Australian dataset, respectively. It is well known that higher the AUC value the better the classifier is said to be. Accordingly, the classifiers are arranged in the descending order of AUC in the tables. As clearly indicated in Table 3, PTVPSO-FKNN outperforms all other methods with the AUC of 81.69%, except the Type II error which is slightly higher than that of PNN. PNN is next to PTVPSO-FKNN with the AUC of 79.89%, Type I error of 21.71%, Type II error of 18.52% and ACC of 79.58%, followed by BPNN, KNN, ELM and SVM. For the Australian dataset whose results are shown in Table 4, we can also observe that PTVPSO-FKNN performs best among all the available methods with the AUC of 87.07%, except the Type I error which is slightly higher than that of SVM. SVM is next to PTVPSO-FKNN with the AUC of 86.08%, Type I error of 9.36%, Type II error of 18.47% and ACC of 85.80%, followed by ELM, KNN, BPNN and PNN. The results are interesting and exciting, which suggests that the FKNN approach can become a promising alternative bankruptcy prediction tool in financial decision-making, where SVM and ANN are known to be the best models [26].

The better performance of the proposed model is owing to the fact that the TVPSO has aided the FKNN classifier to achieve the maximum classification performance by automatically detecting the optimal neighborhood size k and the fuzzy strength parameter m. The detailed results obtained by the proposed method via 10-fold CV are shown in Tables 5 and 6 for the Wieslaw dataset and the Australian dataset, respectively. As shown in the two tables, it can be observed that the values of k and m are different for each fold of the data. With the optimal combination of k and m, FKNN obtained different best classification performance in each fold in terms of the ACC, Type I error, Type II error and AUC. In addition,

Table 3
The ACC, Type I error, Type II error and AUC achieved by different classifiers on the Wieslaw dataset.

Classifiers	ACC (%)	Type I error (%)	Type II error (%)	AUC (%)
PTVPSO-FKNN	**81.67**	**17.58**	19.04	**81.69**
PNN	79.58	21.71	**18.52**	79.89
BPNN	77.92	20.84	21.46	78.71
KNN	78.75	21.46	21.39	78.57
ELM	77.50	19.11	23.97	78.46
SVM	76.67	18.96	26.55	77.26

The best value is shown in bold.

Table 4
The ACC, Type I error, Type II error and AUC achieved by different classifiers on the Australian dataset.

Classifiers	ACC (%)	Type I error (%)	Type II error (%)	AUC (%)
PTVPSO-FKNN	**87.10**	13.21	**12.66**	**87.07**
SVM	85.80	**9.36**	18.47	86.08
ELM	85.65	12.89	15.31	85.90
KNN	85.80	12.12	16.25	85.82
BPNN	85.80	14.02	14.56	85.71
PNN	85.42	12.88	16.41	85.36

The best value is shown in bold.

Table 5
The detailed results obtained by TVPSO-FKNN via 10-fold CV on the Wieslaw dataset.

Fold	#k	#m	ACC (%)	Type I error (%)	Type II error (%)	AUC (%)
#1	100	1.27	83.3333	18.18	15.3846	83.2168
#2	55	1.33	83.3333	18.18	15.3846	83.2168
#3	56	1.39	83.3333	16.67	16.6667	83.3333
#4	84	1.29	79.1667	18.18	23.0769	79.3706
#5	38	1.35	79.1667	18.18	23.0769	79.3706
#6	66	1.14	83.3333	16.67	16.6667	83.3333
#7	66	1.34	79.1667	18.18	23.0769	79.3706
#8	1	1.33	79.1667	16.67	25.0000	79.1667
#9	1	3.00	83.3333	18.18	15.3846	83.2168
#10	14	1.27	83.3333	16.67	16.6667	83.3333
Avg.	48.10	1.47	81.67	17.58	19.04	81.69
Dev.	34.05	0.54	2.15	0.78	3.96	2.04

Table 6
The detailed results obtained by TVPSO-FKNN via 10-fold CV on the Australian dataset.

Fold	#k	#m	ACC (%)	Type I error (%)	Type II error (%)	AUC (%)
#1	8	74.99	88.41	22.22	0	88.89
#2	78	97.94	79.71	20.00	20.59	79.71
#3	77	95.73	91.30	12.20	3.57	92.12
#4	99	98.06	84.06	10.81	21.88	83.66
#5	45	61.80	88.41	10.53	12.90	88.29
#6	66	76.89	89.86	9.52	11.11	89.68
#7	9	23.09	91.30	10.81	6.25	91.47
#8	100	100	86.96	13.16	12.90	86.97
#9	71	69.73	85.51	10.00	20.69	84.66
#10	88	32.21	85.51	12.82	16.67	85.26
Avg.	72.10	73.04	87.10	13.21	12.66	87.07
Dev.	27.53	27.50	3.58	4.36	7.56	3.82

according to our preliminary experiment, k and m can be varied automatically when perform another run of 10-fold CV. The explanation lies in the fact that the two parameters are evolved together by the TVPSO algorithm according to the specific distribution of the training data at hand. It indicates that the optimal values of k and m can always be adaptively specified by TVPSO during each run.

5.2. Experiment II: classification with the PTVPSO-FKNN model with feature selection

As described earlier, the proposed PTVPSO-FKNN model aimed at enhancing the FKNN classification process by not only dealing with the parameters optimization but also automatically identifying the subset of the most discriminative features. In this experiment, we attempt to explore the capability of the PTVPSO-FKNN to further enhance the performance of the FKNN classifier by using the TVPSO. Tables 7 and 8 list the results of PTVPSO-FKNN with and without feature selection for the Wieslaw dataset and the Australian dataset respectively. As shown in Table 7, results obtained on the Wieslaw dataset using PTVPSO-FKNN with feature selection significantly outperforms PTVPSO-FKNN without feature selection in terms of Type I error, Type II error, AUC and ACC at the statistical significance level of 0.05. On the Australian dataset, PTVPSO-FKNN with feature selection significantly outperforms PTVPSO-FKNN without feature selection in terms of Type I error, AUC and ACC at the statistical significance level of 0.1. By using feature selection, the ACC, AUC, Type I error and Type II error have been improved by 2.5%, 2.55%, 1.71% and 3.38% on the Wieslaw dataset, and by 2.47%, 2.74%, 4.03% and 1.47% on the Australian dataset, respectively. For comparison purpose, we conducted the comparative study between TVPSO based and GA based FKNN on the two datasets as shown in Tables 9 and 10. From Table 9, it can be seen that PTVPSO-FKNN outperforms GA-FKNN in terms of Type I error, AUC and ACC on the Wieslaw dataset, though the difference between them is not statistically significant. For the Australian dataset, PTVPSO-FKNN significantly outperforms GA-FKNN in terms of AUC and ACC at the significant level of 0.1, and achieves better performance in terms of Type I error and Type II error as shown in Table 10. From the tables, we can also find that PTVPSO-FKNN has achieved better performance with a smaller feature subset than GA-FKNN on both datasets under investigation. Moreover, during the evolving process, we also observe that the convergence speed of TVPSO is faster than that of GA, and GA is more time-consuming than TVPSO as well. It reflects that TVPSO has stronger search ability than GA on the tested problems. In addition, it is interesting to see that the standard deviation for the acquired performance by the PTVPSO-FKNN is much smaller than that of GA-FKNN on both datasets, which indicates consistency and stability of the proposed model.

To explore how many features and what features are selected during the PSO feature selection procedure, we attempted to further investigate the detail of the feature selection mechanism of the PSO algorithm. For simplicity, here we only took the Wieslaw dataset for example. The original numbers of features of the dataset is 30. As shown in Table 11, not all features are selected for classification after the feature selection. Furthermore, feature selection

Table 7
Experimental results of PTVPSO-FKNN with and without feature selection (%) on the Wieslaw dataset.

Performance metric	PTVPSO-FKNN without feature selection	PTVPSO-FKNN with feature selection	Paired t-test p-value
Type I error	17.58 ± 0.78	15.87 ± 2.42	0.043
Type II error	19.04 ± 3.96	15.66 ± 1.94	0.020
AUC	81.69 ± 2.04	84.24 ± 1.75	0.003
ACC	81.67 ± 2.15	84.17 ± 1.76	0.005

Table 8
Experimental results of PTVPSO-FKNN with and without feature selection (%) on the Australian dataset.

Performance metric	PTVPSO-FKNN without feature selection	PTVPSO-FKNN with feature selection	Paired t-test p-value
Type I error	13.21 ± 4.36	9.18 ± 4.49	0.090
Type II error	12.66 ± 7.56	11.19 ± 3.58	0.490
AUC	87.07 ± 0.04	89.81 ± 2.27	0.053
ACC	87.10 ± 0.04	89.57 ± 2.25	0.090

Table 9
Experimental results of PTVPSO-FKNN vs. GA-FKNN (%) on the Wieslaw dataset.

Performance metric	PTVPSO-FKNN	GA-FKNN	Paired t-test p-value
Type I error	15.87 ± 2.42	17.02 ± 5.08	0.354
Type II error	15.66 ± 1.94	14.94 ± 10.47	0.544
AUC	84.24 ± 1.75	84.02 ± 4.20	0.450
ACC	84.17 ± 1.76	83.33 ± 3.40	0.443
Selected features	15.30 ± 2.75	16.00 ± 3.13	0.010

Table 10
Experimental results of PTVPSO-FKNN vs. GA-FKNN (%) on the Australian dataset.

Performance metric	PTVPSO-FKNN	GA-FKNN	Paired t-test p-value
Type I error	9.18 ± 4.49	11.50 ± 4.62	0.138
Type II error	11.19 ± 3.58	12.32 ± 5.36	0.455
AUC	89.81 ± 2.27	88.09 ± 2.77	0.079
ACC	89.57 ± 2.25	87.97 ± 2.46	0.075
Selected features	9.50 ± 1.58	10.30 ± 1.70	0.011

Table 11
The subset of features selected by PTVPSO-FKNN via 10-fold CV on the Wieslaw dataset.

Fold	Selected features
#1	$X_2X_4X_5X_7X_{10}X_{11}X_{12}X_1\ {}_5X_{20}X_{22}X_{23}X_{26}X_{27}$
#2	$X_1X_3X_4X_6X_7X_8X_{11}X_{13}\ X_{15}X_{16}X_{17}X_{18}X_{19}X_{20}X_{23}\ X_{25}X_{30}$
#3	$X_1X_2X_4X_6X_7X_9X_{13}X_{16}\ X_{20}X_{22}X_{23}X_{24}X_{25}X_{27}$
#4	$X_1X_2X_3X_4X_5X_9X_{10}X_{12}\ X_{13}X_{15}X_{17}X_{18}X_{20}X_{22}X_{23}\ X_{24}X_{25}X_{29}$
#5	$X_1X_2X_3X_6X_7X_8X_9X_{10}\ X_{11}X_{12}X_{15}X_{18}X_{19}X_{20}X_{23}X\ {}_{25}X_{27}X_{28}X_{29}X_{30}$
#6	$X_5X_7X_9X_{14}X_{17}X_{18}X_{19}X\ {}_{21}X_{23}X_{24}X_{25}X_{27}X_{30}$
#7	$X_2X_4X_5X_7X_8X_{12}X\ {}_{13}X_{16}X_{17}X_{18}X_{21}X_{23}\ X_{25}X_{29}X_{30}$
#8	$X_1X_2X_3X_4X_5\ X_7X_8X_{16}X_{19}X_{20}X\ {}_{25}X_{27}X_{29}$
#9	$X_1X_5X_9X_{12}X_{16}X\ {}_{18}X_{20}X_{23}X_{24}X_{25}X_{26}X_{28}$
#10	$X_1X_2X_5X_8X_9X_{10}X_{11}\ X_{14}X_{15}X_{16}X_{17}X_{18}X_{21}X_{23}X\ {}_{25}X_{27}X_{28}X_{30}$

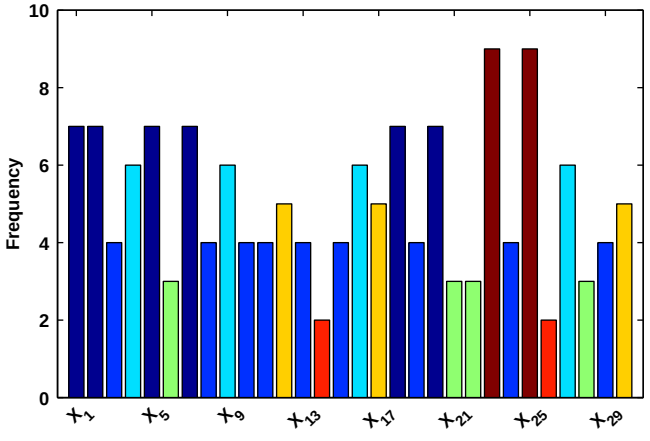

Fig. 4. The frequency of the selected features via 10-fold CV on the Wieslaw dataset.

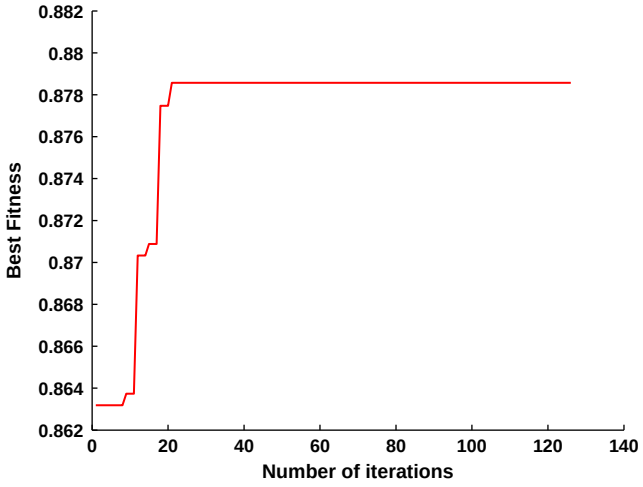

Fig. 5. The best fitness during the training stage for fold #1 on the Wieslaw dataset.

has increased the classification accuracy, as demonstrated in Tables 7 and 8. The average number of selected features by PTVPSO-FKNN is 15.3, and its most important features are C/CL (*X*1), C/TA (*X*2), CA/TA (*X*4), WC/TA (*X*5), S/I (*X*7), NP/TA (*X*9), S/R2(*X*16), S/CA (*X*18), S/TA2(*X*20), R/L (*X*23), L/TA (*X*25) and LTL/E (*X*27), which can be found in the frequency of the selected features of 10-fold CV as shown in Fig. 4. Note that the important features (financial ratios) selected by the proposed model are indeed important from the knowledge perspective also as they are related to current liabilities and long term liabilities, current assets, shareholders' equity and cash, sales, inventory, working capital, net profit, receivables, liabilities, total assets.

To observe the evolutionary process in PTVPSO-FKNN, Fig. 5 shows the evolution of the best fitness for fold 1# within 10-fold CV on the Wieslaw dataset. It should be noted that these results are calculated based on the global best positions, namely, the fitness of all the local best positions on the training set are calculated to obtain the best fitness of the population in each generation. The evolutionary processes are quite interesting. It can be observed that the fitness curves gradually improved from iteration 1 to 130 and exhibited no significant improvements after iteration 22, eventually stopped at the iteration 130 where the particles reached the stopping criterion (100 successively same *gbest* values). The fitness increase rapidly in the beginning of the evolution, after certain number of generations, it starts increasing slowly. During the latter part of the evolution, the fitness keeps stability until the stopping criterion is satisfied. This demonstrates that PTVPSO-FKNN can converge quickly toward the global optima, and fine tune the solutions very efficiently. The phenomenon illustrates the effectiveness of PTVPSO-FKNN in simultaneously evolving the parameters (k and m) and the features through using the TVPSO algorithm.

5.3. Experiment III: comparison between the parallel TVPSO-FKNN model and the serial one

In order to reduce further the running time of the serial TVPSO-FKNN model, we implemented the TVPSO-FKNN model in a parallel environment. To validate the efficiency of the parallel version, here we attempted to compare the performance of the PTVPSO-FKNN with that of TVPSO-FKNN. Table 12 reports the average results of Type I error, Type II error, AUC, ACC and computational time in seconds via 10-fold CV using two models on the two datasets. It can be seen that PTVPSO-FKNN and TVPSO-FKNN give al-

Table 12
The performance of PTVPSO-FKNN and TVPSO-FKNN.

Datasets	PTVPSO-FKNN					TVPSO-FKNN				
	Type I error (%)	Type II error (%)	AUC (%)	ACC (%)	CPU Time (s)	Type I error (%)	Type II error (%)	AUC (%)	ACC (%)	CPU time (s)
Wieslaw	15.87 ± 2.42	15.66 ± 1.94	84.24 ± 1.75	84.17 ± 1.76	120.21 ± 23.34	15.53 ± 5.47	15.95 ± 1.87	84.26 ± 1.98	84.20 ± 1.55	379.56 ± 15.24
Australian	9.18 ± 4.49	11.19 ± 3.58	89.81 ± 2.27	89.57 ± 2.25	208.31 ± 4.74	8.87 ± 4.43	11.82 ± 3.58	89.66 ± 2.57	89.57 ± 2.63	681.26 ± 12.54

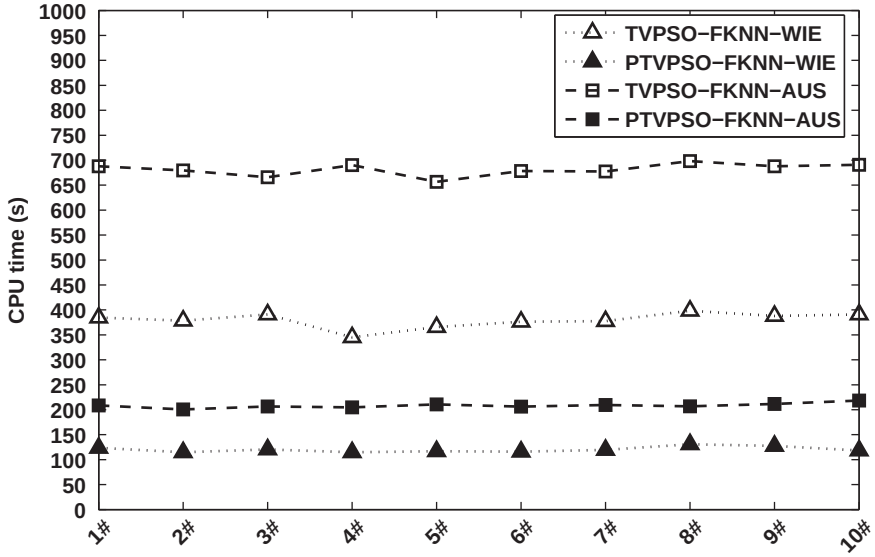

Fig. 6. The average CPU time costs of two models via 10-fold CV on the Wieslaw dataset and Australian dataset (the legend TVPSO-FKNN-WIE and PTVPSO-FKNN-WIE represent the serial model and the parallel model performing on the Wieslaw dataset, respectively, TVPSO-FKNN-AUS and PTVPSO-FKNN-AUS represent the serial model and the parallel model performing on the Australian dataset, respectively).

most the same results on both datasets, minor difference between the parallel model and the serial one is attributed to different partitions of the data are chosen when perform different folds within 10-fold CV. Thus, it verifies the correctness of the parallel design and implementation.

As shown in the Table 12, it can be seen that the average training time within the 10-fold CV for the TVPSO-FKNN was about 3.2 times that of the PTVPSO-FKNN on the Wieslaw dataset, while about 3.3 times that of PTVPSO-FKNN on the Australian dataset. Moreover, the average CPU time spent by the two methods within 10-fold CV has been presented in Fig. 6. It can be observed that PTVPSO-FKNN cost much fewer CPU time than TVPSO-FKNN on each fold of the dataset. It indicates that the TVPSO-FKNN has benefited a great deal from the parallel implementation with respect to the computational time. It is worth noticing that here only a quad-core processor is used in this experiment, thus the computational time will be further reduced with increase of the cores.

6. Conclusions and future work

This study provides a novel model for bankruptcy prediction. The main novelty of this model is in the proposed TVPSO-based approach, which aims at aiding the FKNN classifier to achieve the maximum classification performance. On the one hand, the continuous TVPSO is employed to adaptively specify the two important parameters k and m of the FKNN classifier. On the other hand, the binary TVPSO is adopted to identify the most discriminative features. Moreover, both the continuous and binary TVPSO are implemented in a parallel environment to reduce further the computational time. The experimental results demonstrate that the developed model performs significantly better than the other five state-of-the-art classifiers (KNN, SVM, BPNN, PNN and ELM) in financial application field in terms of Type I error, Type II error, ACC and AUC on two real-life cases. In addition, the experiment reveals that the PTVPSO-FKNN is also a powerful feature selection tool which has detected a subset of best discriminative financial ratios that are really important from the knowledge perspective. Last but not least, the proposed model computes rather efficiently owing to the high performance computing technology.

Hence, it can be safely concluded that, the developed PTVPSO-FKNN model can serve as a promising alternative early warning system in financial decision-making. Meanwhile, we should note that the proposed model does perform efficiently on the data at hand; however, it is not obvious that the parallel algorithm will lead to significant improvement when applying to the financial data with larger instances. Future investigation will pay much attention to evaluating the proposed model in the larger datasets.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 60873149, 60973088, 60773099 and the National High-Tech Research and Development Plan of China under Grant Nos. 2006AA10Z245, 2006AA10A309. This work is also supported by the Open Projects of Shanghai Key Laboratory of Intelligent Information Processing in Fudan University under the Grant No. IIPL-09-007, the Open Project Program of the National Laboratory of Pattern Recognition (NLPR) and the basic scientific research fund of Chinese Ministry of Education.

References

[1] W.H. Beaver, Financial ratios as predictors of failure, Journal of Accounting Research 4 (1966) 71–111.
[2] E.I. Altaian, Financial ratios, discriminant analysis and the prediction of corporate bankruptcy, Journal of Finance 23 (4) (1968) 589–609.
[3] J.A. Ohlson, Financial ratios and the probabilistic prediction of bankruptcy, Journal of Accounting Research (1980) 109–131.
[4] R.C. West, A factor-analytic approach to bank condition, Journal of Banking & Finance 9 (2) (1985) 253–266.
[5] M.D. Odom, R. Sharda, A neural network model for bankruptcy prediction. in: Proceedings of the IEEE International Joint Conference on Neural Networks. San Diego, CA, vol. 2, 1990, pp. 163–168.
[6] R.L. Wilson, R. Sharda, Bankruptcy prediction using neural networks, Decision Support Systems 11 (5) (1994) 545–557.
[7] A.F. Atiya, Bankruptcy prediction for credit risk using neural networks: A survey and new results, IEEE Transactions on Neural Networks 12 (4) (2001) 929–935.
[8] C.F. Tsai, Financial decision support using neural networks and support vector machines, Expert Systems 25 (4) (2008) 380–393.
[9] G. Zhang, Artificial neural networks in bankruptcy prediction: general framework and cross-validation analysis, European Journal of Operational Research 116 (1) (1999) 16–32.
[10] M. Leshno, Y. Spector, Neural network prediction analysis: the bankruptcy case, Neurocomputing 10 (2) (1996) 125–147.
[11] P. Ravisankar, V. Ravi, Financial distress prediction in banks using Group Method of Data Handling neural network, counter propagation neural network and fuzzy ARTMAP, Knowledge-Based Systems 23 (8) (2010) 823–831.
[12] T.E. McKee, A mathematically derived rough set model for bankruptcy prediction, in: C.E. Brown (Ed.), Collected Papers of the Seventh Annual Research Workshop on Artificial Intelligence and Emerging Technologies in Accounting, Auditing and Tax, Artificial Intelligence/Emerging Technologies Section of the American Accounting Association, 1998.
[13] T.E. McKee, T. Lensberg, Genetic programming and rough sets: a hybrid approach to bankruptcy classification, European Journal of Operational Research 138 (2) (2002) 436–451.
[14] A.I. Dimitras, R. Slowinski, R. Susmaga, C. Zopounidis, Business failure prediction using rough sets, European Journal of Operational Research 114 (2) (1999) 263–280.
[15] K.S. Shin, T.S. Lee, H.J. Kim, An application of support vector machines in bankruptcy prediction model, Expert Systems with Applications 28 (1) (2005) 127–135.
[16] J.H. Min, Y.C. Lee, Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters, Expert Systems with Applications 28 (4) (2005) 603–614.
[17] Fengyi Lin, Ching-Chiang Yeh, Meng-Yuan Lee, The use of hybrid manifold learning and support vector machines in the prediction of business failure, Knowledge-Based Systems 24 (1) (2011) 95–101.
[18] A.Y.N. Yip, Predicting business failure with a case-based reasoning approach. in: M.G. Negoita, R.J. Howlett, L.C. Jain (Eds.), Proceedings 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems: KES 2004, Wellington, New Zealand, September 3215/2004, Part III, 2004, pp. 20–25.
[19] C.-S. Park, I. Han, A case-based reasoning with the feature weights derived by analytic hierarchy process for bankruptcy prediction, Expert Systems with Applications 23 (3) (2002) 255–264.
[20] H. Bian, L. Mazlack, Fuzzy-rough nearest-neighbor classification approach, in: Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2003), Chicago, 2003, pp. 500–505.
[21] S. Sarkar, R.S. Sriram, Bayesian models for early warning of bank failures, Management Science 47 (11) (2001) 1457–1475.
[22] L. Sun, P.P. Shenoy, Using Bayesian networks for bankruptcy prediction: some methodological issues, European Journal of Operational Research 180 (2) (2007) 738–753.
[23] Tolga Aydın, Halil Altay Güvenire, Modeling interestingness of streaming association rules as a benefit-maximizing classification problem, Knowledge-Based Systems 22 (1) (2009) 85–99.
[24] Juan L. Castro, Maria Navarro, Jos M. Sànchez, Jos M. Zurita, Introducing attribute risk for retrieval in case-based reasoning, Knowledge-Based Systems 24 (2) (2011) 257–268.
[25] A. Verikas, Z. Kalsyte, M. Bacauskiene, A. Gelzinis, Hybrid and ensemble-based soft computing techniques in bankruptcy prediction: a survey, Soft Computing 14 (9) (2010) 995–1010.
[26] P.R. Kumar, V. Ravi, Bankruptcy prediction in banks and firms via statistical and intelligent techniques – A review, European Journal of Operational Research 180 (1) (2007) 1–28.
[27] J.M. Keller, M.R. Gray, J.A. Givens, A fuzzy k-nearest neighbours algorithm, IEEE Transactions on Systems Man and Cybernetics 15 (4) (1985) 580–585.
[28] J. Sim, S.Y. Kim, J. Lee, Prediction of protein solvent accessibility using fuzzy k-nearest neighbor method, Bioinformatics 21 (12) (2005) 2844.
[29] Y. Huang, Y. Li, Prediction of protein subcellular locations using fuzzy k-NN method, Bioinformatics 20 (1) (2004) 21–28.
[30] T.L. Zhang, Y.S. Ding, K.C. Chou, Prediction protein structural classes with pseudo-amino acid composition: approximate entropy and hydrophobicity pattern, Journal of Theoretical Biology 250 (1) (2008) 186–193.
[31] T.W. Liao, D. Li, Two manufacturing applications of the fuzzy K-NN algorithm, Fuzzy Sets and Systems 92 (3) (1997) 289–304.
[32] S. Yu, S. De Backer, P. Scheunders, Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery, Pattern Recognition Letters 23 (1-3) (2002) 183–190.

[33] H. Bian, L. Mazlack, Fuzzy-rough nearest-neighbor classification approach, in: Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2003), Chicago, 2003, pp. 500–505.
[34] C.F. Tsai, Feature selection in bankruptcy prediction, Knowledge-Based Systems 22 (2) (2009) 120–127.
[35] S.H. Min, J. Lee, I. Han, Hybrid genetic algorithms and support vector machines for bankruptcy prediction, Expert Systems with Applications 31 (3) (2006) 652–660.
[36] V. Ravi, C. Pramodh, Threshold accepting trained principal component neural network and feature subset selection: application to bankruptcy prediction in banks, Applied Soft Computing 8 (4) (2008) 1539–1548.
[37] P. du Jardin, Predicting bankruptcy using neural networks and other classification methods: the influence of variable selection techniques on model accuracy, Neurocomputing 73 (10–12) (2010) 2047–2060.
[38] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, The Journal of Machine Learning Research 3 (2003) 1157–1182.
[39] A. Trabelsi, A. Esseghir Mohamed, New evolutionary bankruptcy forecasting model based on genetic algorithms and neural networks, in: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, 2005, p. 241–245.
[40] B. Back, T. Laitinen, K. Sere, Neural networks and genetic algorithms for bankruptcy predictions, Expert Systems with Applications 11 (4) (1996) 407–413.
[41] J.P. Ignizio, J.R. Soltys, Simultaneous design and training of ontogenic neural network classifiers, Computers & Operations Research 23 (6) (1996) 535–546.
[42] L.H. Chen, H.D. Hsiao, Feature selection to diagnose a business crisis by using a real GA-based support vector machine: an empirical study, Expert Systems with Applications 35 (3) (2008) 1145–1155.
[43] C.H. Wu, G.H. Tzeng, Y.J. Goo, W.C. Fang, A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy, Expert Systems with Applications 32 (2) (2007) 397–408.
[44] H. Yi-Chung, Incorporating a non-additive decision making method into multi-layer neural networks and its application to financial distress analysis, Knowledge-Based Systems 21 (5) (2008) 383–390.
[45] H. Yi-Chung, T. Fang-Mei, Functional-link net with fuzzy integral for bankruptcy prediction, Neurocomputing 70 (16–18) (2007) 2959–2968.
[46] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Sixth International Symposium on Micro Machine and Human Science, Nagoya, 1995, pp. 39–43.
[47] B. Chapman, G. Jost, R. Van der Pas, Using OpenMP: Portable Shared Memory Parallel Programming, The MIT Press, 2007.
[48] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Network, vol. 4, 1995, 1942–1948.
[49] R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources. in: Proceedings of 2001 Congress on evolutionary computation, vol. 1, 2001, pp. 81–86.
[50] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, NJ, 1998, pp. 69–73.
[51] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, 1999: Congress on evolutionary computation, Washington DC, USA, pp. 1945–1949.
[52] A. Ratnaweera, S. Halgamuge, H. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 240–255.
[53] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of IEEE conference on systems, man and cybernetics, 1997, pp. 4104–4108.
[54] P. Wieslaw, Application of discrete predicting structures in an early warning expert system for financial distress. 2004, Ph.D. Thesis, Szczecin Technical University, Szczecin.
[55] W. Pietruszkiewicz, Dynamical systems and nonlinear Kalman filtering applied in classification, in: 7th IEEE International Conference on Cybernetic Intelligent Systems, 2008, pp. 1–6.
[56] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley, New York, 2001.
[57] C.W. Hsu, C.C. Chang, C.J. Lin, A practical guide to support vector classification, 2003, Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003. <http://www.csie.ntu.edu.tw/cjlin/libsvm/>.
[58] S.L. Salzberg, On comparing classifiers: pitfalls to avoid and a recommended approach, Data Mining and Knowledge Discovery 1 (3) (1997) 317–328.
[59] A. Statnikov, I. Tsamardinos, Y. Dosbayev, C.F. Aliferis, GEMS: A system for automated cancer diagnosis and biomarker discovery from microarray gene expression data, International Journal of Medical Informatics 74 (7-8) (2005) 491–503.
[60] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[61] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[62] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines. 2001. <http://www.csie.ntu.edu.tw/cjlin/libsvm>.
[63] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27 (8) (2006) 861–874.