An Adaptive Fuzzy k-Nearest Neighbor Method Based on Parallel Particle Swarm Optimization for Bankruptcy Prediction

Hui-Ling Chen, Da-You Liu, Bo Yang, Jie Liu, Gang Wang, and Su-jing Wang

Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of EducationJilin University, Changchun 130012, China
chenhuiling.jlu@gmail.com; {liudy*, ybo, liu_jie}@jlu.edu.cn; wanggang.jlu@gmail.com; wangsj08@mails.jlu.edu.cn

Abstract. This study proposes an efficient non-parametric classifier for bankruptcy prediction using an adaptive fuzzy k-nearest neighbor (FKNN) method, where the nearest neighbor k and the fuzzy strength parameter m are adaptively specified by the particle swarm optimization (PSO) approach. In addition to performing the parameter optimization for FKNN, PSO is utilized to choose the most discriminative subset of features for prediction as well. Time varying acceleration coefficients (TVAC) and inertia weight (TVIW) are employed to efficiently control the local and global search ability of PSO. Moreover, both the continuous and binary PSO are implemented in parallel on a multi-core platform. The resultant bankruptcy prediction model, named PTVPSO-FKNN, is compared with three classification methods on a real-world case. The obtained results clearly confirm the superiority of the developed model as compared to the other three methods in terms of Classification accuracy, Type I error, Type II error and AUC (area under the receiver operating characteristic (ROC) curve) criterion. It is also observed that the PTVPSO-FKNN is a powerful feature selection tool which has indentified a subset of best discriminative features. Additionally, the proposed model has gained a great deal of efficiency in terms of CPU time owing to the parallel implementation.

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

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. A fair amount of models has been developed for bankruptcy prediction. These models have progressed from statistical methods to the artificial intelligence (AI) approach. A number of statistical methods such as the simple univariate analysis, multivariate discriminant analysis technique, logistic regression approach and factor analysis technique have been typically used for financial applications including bankruptcy prediction. Recent studies in the AI approach, such as artificial neural networks (ANN) , rough set theory , support vector machines (SVM) , k-nearest neighbor method (KNN) and Bayesian network models have also been successfully applied to bankruptcy prediction (see [1][2]). 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 with 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. In this way, once the class has been assigned, there is no indication of the significance of membership to indicate how much the instance belongs to a particular class. An improvement over the standard KNN classifier is the Fuzzy k-nearest neighbor classifier (FKNN) [3], which uses concepts from fuzzy logic to assign degree of membership to different classes while considering the distance of its k nearest neighbors. The FKNN method has been frequently used for the classification of biological data, image data and so on. Nevertheless, only few works have paid attention to using FKNN to classify the financial data. Bian et al. [4] used FKNN as a reference classifier in their experiments

* Corresponding author

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 nearest neighbors k and the fuzzy strength parameter m, which play a significant role in improving the prediction power for FKNN. This study aims to 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 [5], 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, eliminate the irrelevant of dispensable features and build a good learning model. In bankruptcy prediction, genetic algorithms (GA) are usually used to select a subset of input features or to find appropriate hyper-parameter values of a predictor. Compared with GA, particle swarm optimization (PSO) algorithm has no crossover and mutation operators, it is simple and computationally inexpensive both in memory and runtime. 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 nearest neighbor 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 time variant 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 [6]. The efficacy of the proposed bankruptcy prediction model PTVPSO-FKNN is compared with three reference classification methods on a real-world case. All these classifiers are compared with respect to the classification accuracy, Type I error, Type II error and the AUC (area under the receiver operating characteristic (ROC) curve) criterion. 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 model. 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 fuzzy k-nearest neighbor method (FKNN) and particle swarm optimization algorithm (PSO). 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 are summarized in Section 6.

2 Background materials

2.1 Fuzzy k-Nearest Neighbor Algorithm (FKNN)

The k-nearest neighbor algorithm (KNN) is one of the oldest and simplest non parametric pattern classification methods [7]. 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) [3]. 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:

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

where $i = 1, 2, \cdots, c$, and $j = 1, 2, \cdots, k$, 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 Euclidean distance between x and its jth

nearest neighbor x_j. And 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 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:

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

The value n_j is the number of neighbors found which belong to the jth class. In our experiments, we have found that the second way lead 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.

2.2 Time Variant Particle Swarm Optimization (TVPSO)

Particle swarm optimization (PSO) was first developed by Kennedy and Eberhart [8]. 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:

$$v_{i,j}^{n+1} = w \times v_{i,j}^{n} + c_1 \times r_1(p_{i,j}^{n} - x_{i,j}^{n}) + c_2 \times r_2(p_{g,j}^{n} - x_{i,j}^{n}) \tag{3}$$

$$x_{i,j}^{n+1} = x_{i,j}^{n} + v_{i,j}^{n+1}, j \ = 1, 2, \cdots, d \tag{4}$$

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 $[-v_{max}, v_{max}]$. Inertia weight w is updated according to the following equation:

$$w = w_{\min} + (w_{\max} - w_{\min}) \frac{(t_{\max} - t)}{t_{\max}} \tag{5}$$

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. Eq. (5) is also known as Time varying inertia weight (TVIW), which will be incorporated to the TVPSO. c_1 and c_2 are acceleration coefficients, to better balance the search space between the global exploration and local exploitation, Time varying acceleration coefficients (TVAC) have been introduced in [9]. This concept will be adopted in this study to ensure the better search for the solutions. The core idea of TVAC is that c_1 decreases from its initial value of c_{1i} to c_{1f}, while c_2 increases from c_{2i} to c_{2f} using the following equations as in [9]. TVAC can be mathematically represented as follows:

$$c_1 = (c_{1f} - c_{1i}) \frac{t}{t_{\max}} + c_{1i} \tag{6}$$

$$c_2 = (c_{2f} - c_{2i}) \frac{t}{t_{\max}} + c_{2i} \tag{7}$$

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, one discrete PSO version introduced by Kennedy and Eberhart [10] was employed to act as the feature selection tool. In the binary PSO, A sigmoid function is applied to transform the velocity from continuous space to probability space:

$$sig(v_{i,j}) = \frac{1}{1 + \exp(-v_{i,j})}, j \ = 1, 2, \ldots, d \tag{8}$$

The velocity update Eq. (3) keeps 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:

$$x_{ij}^{n+1} = \begin{cases} 1, & \text{if } rnd < sig(v_{i,j}) \\ 0, & \text{if } rnd \geq sig(v_{i,j}) \end{cases}, j = 1, 2, \cdots, d \tag{9}$$

where $sig(v_{i,j})$ is calculated according to Eq. (8), and rand is a uniform random number in the range $[0, 1]$.

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 nearest neighbor 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 time variant PSO are combined together to dynamically conduct the parameter optimization and feature selection. The obtained appropriate feature subset can served as the input into the FKNN classifier to conduct the classification task. Here, we first describe the model based on the serial PSO algorithm, termed TVPSO-FKNN, and then implement it in parallel.

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 with the selected feature vector in Step 2.
- Step 4: It is well known that higher the AUC value the better the classifier is said to be. And 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 fitness function, The fitness value was calculated according to the following objective function:

$$\begin{cases} f_1 = \text{AUC} \\ f_2 = (1 - \frac{\sum_{j=1}^{n} f_{ti}}{n}) \\ f = \alpha \times f_1 + \beta \times f_2 \end{cases} \tag{10}$$

  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, f_{ti} 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 sub-objective function. In the function f, variable 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 more depend on the classification accuracy, hence the α value is set as much bigger than that of β. According to our preliminary experiments, the value 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.(3-4) and the features using Eq.(3), Eqs.(8-9) 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. (10). Notice that PSO is used for optimization tasks where the nearest neighbor 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 Model on the Multi-Core Platform (PTVPSO-FKNN)

In this section, we put forward a parallel implementation of TVPSO-FKNN model which is performed on multi-core processor by using OpenMP. The architecture of the multi-core platform is divided into three lays as shown in Fig. 2(a): 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: It 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 TVPSO-FKNN is summarized in Algorithm 1.

Algorithm 1 PTVPSOFKNN

```
Initialize system parameters.
Train FKNN model.
//current number of iteration (cni), maximum number of iteration (mni)
while cni < mni do
  for each particle do
    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
```

4 Experimental Design

4.1 Data Description

The financial data used for this study was taken from Wieslaw [11] 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

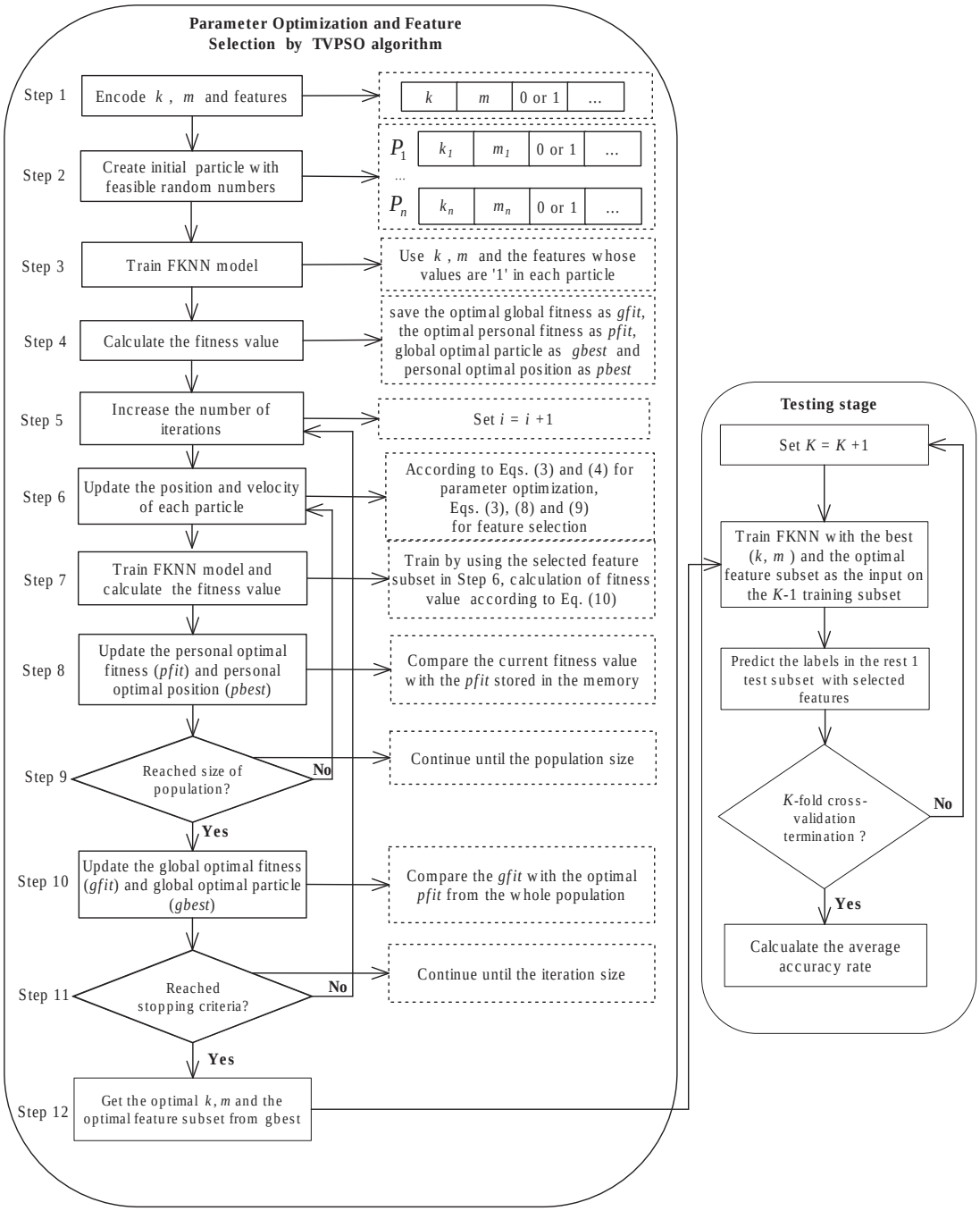
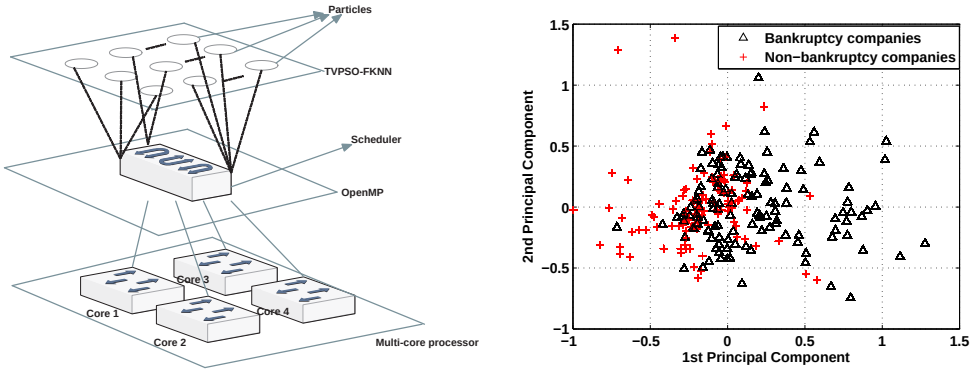


Fig. 1. Overall procedure of the TVPSO-FKNN model



(a) (b)

Fig. 2. (a)The architecture of parallel running environment of TVPSO-FKNN. (b)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.

years before bankruptcy toke place. It should be noted that the size of the data set is not that large compared to the majority of bankruptcy prediction studies. However, according to [12], the dataset is reliable since increasing the dataset length does not lead to the accuracy increase. Fig. 2(b) 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 (PCA) algorithm. As shown in this figure, there is apparently strong overlap between the bankrupt companies and non-bankrupt ones.

Data was normalized by scaling them into the interval of $[-1, 1]$. In order to gain an unbiased estimate of the generalization accuracy, the k-fold CV presented by Salzberg [13] 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.

4.2 Experimental Setup

The proposed PTVPSO-FKNN model was implemented using Visual C++ 2008 and OpenMP. For SVM, LIBSVM implementation is utilized, which was originally developed by Chang and Lin [14]. We implemented the PSO algorithm, FKNN and KNN from scratch. The MLP was created, trained and implemented using Matlab neural network toolbox with BP and the training algorithm of Levenberg-Marquardt. 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 was set as follows. The number of the iterations and particles was set to 250 and 8, respectively. The searching ranges for k and m are as follows: $k \in [1, 100]$ and $m \in [1, 10]$. v_{max} was set about 60% of the dynamic range of the variable on each dimension for the continuous type of dimensions. Therefore, $[-v_{max}, v_{max}]$ was predefined as [0.6, 60] for parameter k, and as $[0.6, 6]$ for parameter m. For the discrete type particle for feature selection, $[-v_{max}, v_{max}]$ was set as $[-6, 6]$. As suggested in [9],c_{1i},c_{1f},c_{2i} and c_{2f} were 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} were set to 0.9 and 0.4, respectively.

For SVM, we considered the nonlinear SVM based on the popular Gaussian (RBF) kernel, and a grid-search technique [15] was 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 were varied between $C = \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\gamma = \{2^{-15}, 2^{-13}, \ldots, 2^{1}\}$. For KNN, we found the best result was achieved when $k = 1$ by using 10-fold CV. Therefore, we selected $k = 1$ for the subsequent analysis. Concerning MLP, we used the three layer back-propagation network to train ANN. We tried 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. The best result was obtained with the hidden layer of 15 and the learning epoch of 200.

4.3 Measure for Performance Evaluation

Type I error, Type II error, total classification accuracy (ACC) and the area under the Receiver Operating Characteristic curve (AUC) [16] were used to test the performance of the proposed PTVPSO-FKNN model. They were the most widely used measures to assess the performance of bankruptcy prediction systems [1]. Type I and Type II errors were two important measures which described how well the classifier discriminates between case with non-bankruptcy and with bankruptcy. Type I error measures the proportion of bankrupt cases which are incorrectly identified as non-bankrupt ones. Type II error measures the proportion of non-bankrupt cases which are incorrectly identified as bankrupt ones. The receiver operating characteristic (ROC) curve is a graphical display that gives the measure of the predictive accuracy of a logistic model [16]. 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 entire feature space with 30 features (financial ratios). In order to verify the effectiveness of the proposed model, TVPSO-FKNN was compared with three other reference classifiers (SVM, KNN and ANN). Table 1 shows the results achieved with all four investigated classifiers (PTVPSO-FKNN, SVM, KNN and ANN) for the financial data with the form of 'average ± standard deviation'. 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 table. As clearly indicated in the table, PTVPSO-FKNN outperforms all other methods with the classification accuracy of 81.67%, Type I error of 17.58%, Type II error of 19.04% and AUC of 81.69%. MLP is next to PTVPSO-FKNN with classification accuracy of 77.92%, Type I error of 20.84%, Type II error of 21.46% and AUC of 78.71%, followed by KNN and SVM. The superiority of the PTVPSO-FKNN is statistically significant as shown by the paired t-test in Tables (2-3), where the significant level is 5%. The results are interesting and exciting, it 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 [2].

Table 1. The ACC, Type I and Type II errors and AUC achieved with different classifiers

Classifiers	ACC (%)	Type I error (%)	Type II error (%)	AUC (%)
PTVPSO-FKNN	81.67±2.15	17.58±0.78	19.04±3.96	81.69±2.04
SVM	76.67±4.65	18.96±8.46	26.55±7.93	77.26±5.62
KNN	78.75±3.65	21.46±5.07	21.39±4.13	78.57±3.78
MLP	77.92±5.22	20.84±7.21	21.46±9.84	78.71±6.48

Table 2. Paired t-test results of Type I and Type II error

Type I error / Type II error	TVPSO-FKNN t-value (significance)
MLP	-2.243(0.037)/-2.589(0.047)
KNN	-2.332(0.045)/-2.366(0.042)
SVM	-3.045(0.026)/-3.122(0.032)

Negative values indicate that the ith classifier has a Type I error /Type II error higher than that of the jth one.

The better performance of the proposed model can be explained by the fact that the TVPSO has aided the FKNN classifier to achieve the maximum classification performance by automatically detecting the optimal nearest neighbor k and the fuzzy strength parameter m. The detailed values of parameters k and m via 10-fold CV using the proposed model is shown in Table 4. From

Table 3. Paired t-test results of ACC and AUC

ACC / AUC	TVPSO-FKNN t-value (significance)
MLP	-3.345(0.017)/-3.623(0.021)
KNN	-3.280(0.009)/-3.168(0.011)
SVM	-4.458(0.005)/-4.854(0.023)

Negative values indicate that the ith classifier has an ACC/AUC lower than that of the jth one.

Table 4. The detailed parameter values obtained through 10-fold CV

Fold	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
k	23	33	55	14	1	25	86	43	22	15
m	1.27	1.33	1.39	1.35	1.29	1.34	1.67	1.45	1.32	3.00

the table, it can be observed that the values of k and m are different for each fold of the data. And according to our preliminary experiment, they 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 of the experiment. Moreover, it is interesting to see that the standard deviation for the acquired performance by the PTVPSO-FKNN is much smaller than that of the other three classifiers, which indicates consistency and stability of the proposed model.

5.2 Experiment II: Classification Using 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 boost the performance of the FKNN classifier by using the TVPSO. Table 5 lists the best results of PTVPSO-FKNN with and without feature selection for Wieslaw dataset. As shown in this table, results obtained using PTVPSO-FKNN with feature selection significantly outperforms PTVPSO-FKNN without feature selection in terms of the Type I error, Type II error, AUC and classification accuracy at the statistical significance level of 5%. By using feature selection, the classification accuracy, AUC values, Type I error and Type II error have been improved by 2.5%, 2.55%, 1.71% and 3.38% on average, respectively.

Table 5. Experimental results of the PTVPSO-FKNN with and without feature selection(%)

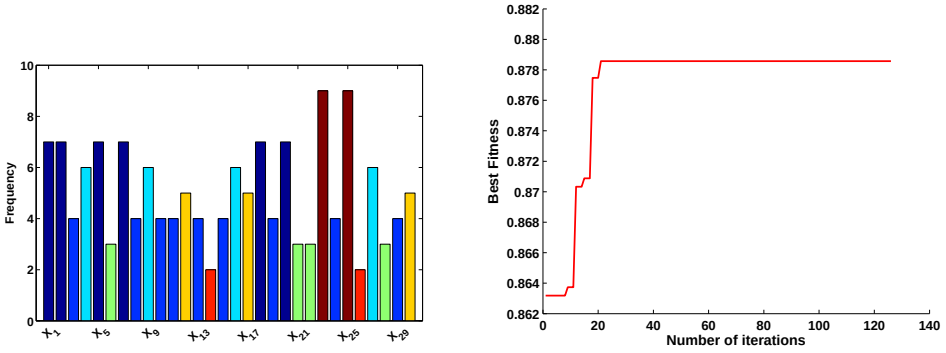
Performance metric	PTVPSO-FKNN without feature selection	PTVPSO-FKNN with feature selection	Paired $t-$test p-value
Type I error	17.58±0.78	15.87±2.42	0.0429
Type II error	19.04±3.96	15.66±1.94	0.0202
AUC	81.69±2.04	84.24±1.75	0.0029
ACC	81.67±2.15	84.17±1.76	0.0051

To explore how many features and what features are selected during the PSO feature selection procedure, we further conducted an experiment on the Wieslaw dataset to investigate the detail of the feature selection mechanism of the PSO algorithm. The original numbers of features of the dataset is 30. As shown in Table 6, not all features are selected for classification after the feature selection. Furthermore, feature selection has increased the classification accuracy, as demonstrated in Table 5. The average number of selected features by PTVPSO-FKNN is 15.3, and its most important features are X_1, X_2, X_4, X_5, X_7, X_9, X_{16}, X_{18}, X_{20}, X_{23}, X_{25} and X_{27}, which can be found in the frequency of the selected features of 10-fold CV as shown in Fig. 3(a).

It should be noticed that 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.

Table 6. The subset of features selected by PTVPSO-FKNN via 10-fold CV

Fold	Selected features
#1	X_2 X_4 X_5 X_7 X_{10} X_{11} X_{12} X_{15} X_{20} X_{22} X_{23} X_{26} X_{27}
#2	X_1 X_3 X_4 X_6 X_7 X_8 X_{11} X_{13} X_{15} X_{16} X_{17} X_{18} X_{19} X_{20} X_{23} X_{25} X_{30}
#3	X_1 X_2 X_4 X_6 X_7 X_9 X_{13} X_{16} X_{20} X_{22} X_{23} X_{24} X_{25} X_{27}
#4	X_1 X_2 X_3 X_4 X_5 X_9 X_{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_1 X_2 X_3 X_6 X_7 X_8 X_9 X_{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_5 X_7 X_9 X_{14} X_{17} X_{18} X_{19} X_{21} X_{23} X_{24} X_{25} X_{27} X_{30}
#7	X_2X_4 X_5 X_7 X_8 X_{12} X_{13} X_{16} X_{17} X_{18} X_{21} X_{23} X_{25} X_{29} X_{30}
#8	X_1 X_2 X_3 X_4 X_5 X_7 X_8 X_{16} X_{19} X_{20} X_{25} X_{27} X_{29}
#9	X_1 X_5 X_9 X_{12} X_{16} X_{18} X_{20} X_{23} X_{24} X_{25} X_{26} X_{28}
#10	X_1 X_2 X_5 X_8 X_9 X_{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}



(a) (b)

Fig. 3. (a) The frequency of the selected features in 10-fold CV on Wieslaw dataset. (b) The best fitness during the training stage for fold #1.

To observe the evolutionary process in PTVPSO-FKNN, Fig. 3(b) shows the evolution of the best fitness for fold 1# during 10-fold CV. 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. It 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 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 on a multi-core platform. 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 7 reported the best results of Type I error, Type II error, ACC, AUC and

the average computational time in seconds using the two models. It can be seen that PTVPSO-FKNN and TVPSO-FKNN give almost the same results, the minor different results between two models may be attributed to different partitions of the data are chosen when perform different runs of 10-fold CV. Thus, it verifies the correctness of the parallel design and implementation. However, the training time for the TVPSO-FKNN was 3.3 times that of the PTVPSO-FKNN, which indicates that the TVPSO-FKNN has benefited a great deal from the parallel implementation with respect to the computational time. Additionally, it should be noted that only a quad-core processor was used in this experiment, thus the computational time will be further reduced with increase of the cores.

Table 7. The performance comparison of PTVPSO-FKNN with TVPSO-FKNN

Performance metric	PTVPSO-FKNN	TVPSO-FKNN
Type I error (%)	15.87±2.42	15.53±2.56
Type II error (%)	15.66±1.94	15.95±1.87
AUC (%)	84.24±1.75	84.26±1.98
ACC (%)	84.17±1.76	84.20±1.55
CPU Time (s)	1150.46±23.34	3796.51±30.45

6 Conclusions

This study provides an attractive model PTVPSO-FKNN 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 three state-of-the-art classifiers (KNN, SVM and MLP) in financial application field in terms of the Type I error, Type II error, ACC and AUC on a real life dataset. Moreover, 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. Furthermore, 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 dataset.

Acknowledgments. 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 Grand 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. Verikas, A., Kalsyte, Z., Bacauskiene, M., Gelzinis, A.: Hybrid and ensemble-based soft computing techniques in bankruptcy prediction: A survey. Soft Computing-A Fusion of Foundations, Methodologies and Applications **14**(9) (2010) 995–1010
2. Ravi Kumar, P., Ravi, V.: Bankruptcy prediction in banks and firms via statistical and intelligent techniques-a review. European Journal of Operational Research **180**(1) (2007) 1–28
3. Keller, J.: A Fuzzy k-Nearest Neighbor Algorithm. IEEE Transactions on Systems, Man, and Cybernetics **15**(4) (1985) 580–585

4. Bian, H., Mazlack, L.: Fuzzy-rough nearest-neighbor classification approach. In: Fuzzy Information Processing Society, 2003. NAFIPS 2003. 22nd International Conference of the North American, IEEE (2003) 500–505
5. du Jardin, P.: 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
6. Chapman, B., Jost, G., Van Der Pas, R.: Using OpenMP: portable shared memory parallel programming. The MIT Press (2008)
7. Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13**(1) (1967) 21–27
8. Kennedy, J., Eberhart, R., et al.: Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks. Volume 4., Perth, Australia (1995) 1942–1948
9. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation **8**(3) (2004) 240–255
10. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm optimization. In: Proc. Conf. Systems, Man, and Cybernetics. (1997) 4104–4108
11. Wieslaw, P.: Application of discrete predicting structures in an early warning expert system for financial distress. PhD thesis, Ph. D. Thesis. Szczecin: Szczecin Technical University (2004)
12. Pietruszkiewicz, W.: Dynamical systems and nonlinear Kalman filtering applied in classification. In: 7th IEEE International Conference on Cybernetic Intelligent Systems, 2008. CIS 2008., IEEE (2008) 1–6
13. Salzberg, S.: On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Mining and Knowledge Discovery **1**(3) (1997) 317–328
14. Chang, C., Lin, C.: LIBSVM: a library for support vector machines, 2001. Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm (2001)
15. Chang, C., Lin, C., Hsu, C.: A practical guide to support vector classification. Department of Computer Science and Information Engineering, National Taiwan University, Taiwan (2003)
16. Fawcett, T.: An introduction to ROC analysis. Pattern recognition letters **27**(8) (2006) 861–874