Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 8 (2008) 1381–1391
Applied Soft Computing
www.elsevier.com/locate/asoc

A distributed PSO–SVM hybrid system with feature selection and parameter optimization

Cheng-Lung Huang [a,*], Jian-Fan Dun [b]

[a] Department of Information Management, National Kaohsiung First University of Science and Technology, 2 Juoyue Road, Nantz District, Kaohsiung 811, Taiwan, ROC
[b] Department of Information Management, Huafan University, Taipei, Taiwan, ROC

Received 12 July 2006; received in revised form 7 August 2007; accepted 16 October 2007
Available online 22 October 2007

Abstract

This study proposed a novel PSO–SVM model that hybridized the particle swarm optimization (PSO) and support vector machines (SVM) to improve the classification accuracy with a small and appropriate feature subset. This optimization mechanism combined the discrete PSO with the continuous-valued PSO to simultaneously optimize the input feature subset selection and the SVM kernel parameter setting. The hybrid PSO–SVM data mining system was implemented via a distributed architecture using the web service technology to reduce the computational time. In a heterogeneous computing environment, the PSO optimization was performed on the application server and the SVM model was trained on the client (agent) computer. The experimental results showed the proposed approach can correctly select the discriminating input features and also achieve high classification accuracy.

Keywords: Particle swarm optimization; Support vector machines; Distributed computing; Web service; Data mining; Feature selection

1. Introduction

Classification and prediction are the two most important tasks in data mining. A bundle of approaches was developed to build classification and prediction models, including the support vector machines (SVM), which were suggested by Vapnik [1], and have recently been used in a range of problems. To build a SVM-based classification model (as well as other classification and prediction models, such as the neural network, decision tree, and regression), feature subset selection is an important issue in building classification systems. It is advantageous to limit the number of input features in a classifier to produce a good predictive and less computationally intensive model [2]. With a small and appropriate feature subset, the rationale for the classification decision can be realized easier. In addition to feature selection, proper model parameter setting can improve the SVM classification accuracy. To design a SVM, one must choose a kernel function, set the kernel parameters and determine a soft margin constant C. The parameters that should be optimized include the penalty parameter C and the kernel function parameters such as the gamma (γ) for the radial basis function (RBF) kernel. Thus, appropriate feature subset selection and model parameter setting have a heavy impact on the classification accuracy [3]. Because feature subset selection influences the appropriate kernel parameters and vice versa [4], obtaining the optimal feature subset and SVM parameters must occur simultaneously. The grid algorithm is an alternative to finding the best C and gamma when using the RBF kernel function [5]; however, this approach does not simultaneously perform feature subset selection. Huang and Wang [3] adopted the genetic algorithm to optimize the feature subset and model parameter selection for the SVM. They achieved a promising result.

Instead of using the genetic algorithm, this study tries a new technology, swarm intelligence: particle swarm optimization (PSO). Proposed by Kennedy and Eberhart [6,7] and inspired by social behavior in nature, PSO is a population-based search algorithm that is initialized with a population of random solutions, called particles. Each particle in the PSO flies through the search space at a velocity that is dynamically adjusted according to its own and its companion's historical behavior. Recently, numerous researches on PSO theories

* Corresponding author. Tel.: +886 7 6011000x4127; fax: +886 7 6011042.
E-mail address: clhuang@ccms.nkfust.edu.tw (C.-L. Huang).

1568-4946/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2007.10.007

or applications have been reported [8–17]. Because particle swarm optimization is powerful, easy to implement, and computationally efficient [7], this study introduces PSO as an optimization technique to simultaneously optimize both the optimal feature subset and SVM parameters. We use two types of PSO: the continuous-valued version and discrete version. The continuous-valued version is used to optimize the best SVM model parameters, while the discrete version is used to search the optimal feature subset. Combining PSO with SVM, this novel hybrid approach is proposed here to optimize the parameters and feature subset simultaneously, without degrading the SVM classification accuracy. Due to the time consuming data mining task, the development of a distributed data mining system is essential. The development of cost-effective web-based interoperable data mining tools must overcome various interoperability issues in heterogeneous environments, namely programming language, platform, object model, and data access [18]. PSO can be easily adopted for parallel processing; therefore, an architecture for the distributed PSO-based SVM classifier is implemented in this study.

This paper is organized as follows. Section 2 describes the related works including the basic SVM and PSO concepts. Section 3 describes the PSO–SVM hybrid system. Section 4 presents the experimental results from a simulated dataset. Section 5 implements the PSO–SVM via a distributed architecture. Section 6 gives remarks and provides a conclusion.

2. Related works

2.1. SVM classifier

This section gives a brief description on SVM [19–21]. Given a training set of instance-label pairs (x_i, y_i), $i = 1, 2, \ldots, m$ where $x_i \in R^n$ and $y_i \in \{+1, -1\}$, the generalized linear SVM finds an optimal separating hyperplane $f(x) = \langle w \cdot x \rangle + b$ by solving the following optimization problem:

$$\begin{aligned} \underset{w,b,\xi}{\text{Minimize}} \quad & \frac{1}{2}\langle w \cdot w \rangle + C\sum_{i=1}^{m}\xi_i \\ \text{Subject to :} \quad & y_i(\langle w \cdot x_i \rangle + b) + \xi_i - 1 \geq 0 \\ & \xi_i \geq 0 \end{aligned} \tag{1}$$

where C is a penalty parameter on the training error, and ξ_i is the non-negative slack variables. SVM finds the hyperplane that provides the minimum number of training errors (i.e., to keep the constraint violation as small as possible). This optimization model can be solved by introducing the Lagrange multipliers α_i for its dual optimization model. After the optimal solution α_i^* is obtained, the optimal hyperplane parameters w^* and b^* can be determined, and the indicator function (classifier) can be written as:

$$\text{sign}(\langle w^* \cdot x \rangle + b^*) \quad \text{or} \quad \text{sign}\left(\sum_{i=1}^{m} y_i \alpha_i^* \langle x_i \cdot x \rangle + b^*\right) \tag{2}$$

The nonlinear SVM maps the training samples from the input space into a higher dimensional feature space via a mapping function Φ. By performing such a mapping, the training samples can be linearly separated by applying the linear SVM formulation. The scalar product $\langle \Phi(x_i) \cdot \Phi(x_j) \rangle$ is calculated directly by computing the kernel function $k(x_i, x_j)$ for given training data in an input space. Radial basis function (RBF) is a common kernel function as the follows.

$$\begin{aligned} & k(x_i, x_j) = \exp\left(-\frac{1}{\sigma^2}||x_i - x_j||^2\right) \quad \text{or} \\ & k(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2) \end{aligned} \tag{3}$$

By introducing the kernel function, the nonlinear SVM classifier has the following forms:

$$\begin{aligned} & \text{sign}\left(\sum_{i=1}^{m} y_i \alpha_i^* \langle \Phi(x) \cdot \Phi(x_i) \rangle + b^*\right) \quad \text{or} \\ & \text{sign}\left(\sum_{i=1}^{m} y_i \alpha_i^* k(x, x_i) + b^*\right) \end{aligned} \tag{4}$$

For the multi-class classifier, there are two major multi-class SVM classification strategies: one-against-all and one-against-one, and the latter is adapted in this study. Given a training set with m training samples (x_i, x_j), $i = 1, 2, \ldots, m$ where $x_i \in R^n$ and $y_i \in \{1, \ldots, k\}$. For "one-against-all" strategy [22], k binary SVM decision functions are constructed. The jth decision function is trained by labeling all of the examples in the jth class with positive labels, and all of the examples not in the jth class with negative labels. A new x is classified into the class which has the largest decision function. For "one-against-one" strategy [23,24], $C_2^k = k(k-1)/2$ classifiers are constructed, and each classifier is trained with two different classes (class c_i vs. class c_j). A new x is classified into the majority class voted by all of the indicator functions.

2.2. Particle swarm optimization

2.2.1. Basic concept of particle swarm optimization

Particle swarm optimization is an evolutionary computation technique. Similar to genetic algorithms, PSO is a population-based optimization tool. It is inspired by social behavior among individuals. Particles (individuals) representing a potential problem solution move through an n-dimensional search space. Each particle i maintains a record of the position of its previous best performance in a vector called *pbest*. The *nbest*, is another "best" value that is tracked by the particle swarm optimizer. This is the best value obtained so far by any particle in that particle's neighborhood. When a particle takes the entire population as its topological neighbors, the best value is a global best and is called *gbest*. All particles can share information about the search space.

Representing a possible solution to the optimization problem, each particle moves in the direction of its best solution and the global best position discovered by any particles in the swarm. Each particle calculates its own velocity and updates its position in each iteration. Let $p_{i,d}$ denote the best previous position encountered by the ith particle. $p_{g,d}$ denotes

the global best position thus far, and t denotes the iteration counter. The current velocity of the dth dimension of the ith particle at time t is

$$v_{i,d}(t) = w \times v_{i,d}(t-1) + c_1 \times \mathrm{rnd}(\,) \times (p_{i,d} - x_{i,d}(t-1)) + c_2 \times \mathrm{rnd}(\,) \times (p_{g,d} - x_{i,d}(t-1)) \quad (5)$$

$$v_{i,d} \in [-v_{\max}, v_{\max}] \quad (6)$$

In the above formula, rnd( ) is a random function in the range [0, 1], positive constant c_1 and c_2 are personal and social learning factors, and w is the inertia weight. Inertia weight was first introduced by Shi and Eberhart [25]. Inertia weight balances the global exploration and local exploitation. The velocity is restricted to the $[-v_{\max}, v_{\max}]$ range in which $v_{\max}$ is a predefined boundary value. The value of $v_{\max}$ determines the resolution of the search regions between the present and target position. Eberhart and Shi [26] suggested that $v_{\max}$ be set at about 10–20% of the dynamic range of the variable in each dimension. The new position of a particle is calculated using the following formula:

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \quad (7)$$

2.2.2. Binary PSO

Many optimization problems are set in a space featuring discrete, qualitative distinctions between variables and between levels of variables. Typical examples include problems which require the ordering or arranging of discrete elements, as in scheduling and routing problems.

Kennedy and Eberhart [27] proposed a binary PSO in which 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.

The velocity formula (5) remains unchanged except that $x_{i,d}$, $p_{i,d}$ and $x_{g,d}$ are integers in {0, 1} and $v_{i,d}$ must be constrained to the interval [0.0, 1.0]. This can be accomplished by introducing a sigmoid function $S(v)$, and the new particle position is calculated using the following rule:

$$\text{if } \mathrm{rnd}(\,) < S(v_{i,d}), \text{ then } x_{i,d} = 1; \text{ else } x_{i,d} = 0; \quad (8)$$

where

$$S(v) = \frac{1}{1 + \mathrm{e}^{-v}} \quad (9)$$

The function $S(v)$ is a sigmoid limiting transformation and rnd( ) is a random number selected from a uniform distribution in [0.0, 1.0].

2.2.3. Feature selection

Agrafiotis and Cedeno [28] fist used PSO to selected input features. They combine the PSO with neural networks applied in the construction of parsimonious quantitative structure-activity relationship models. The subset is assembled by spinning wheel and selecting the features that fall under the wheel's marker. This process is repeated k times, where k is the number of desired features in the model. The selection probability $p_{i,d}$ for dimension d of particle i, is computed using the following equation [28]:

$$p_{i,d} = \frac{x_{i,d}^{\alpha}}{\sum_{d=1}^{n} x_{i,d}^{\alpha}} \quad (10)$$

where $x_{i,d}$ is the fractional coordinates obtained by the particle updating equation (Eq. (7)), α is a scaling factor referred to as selection pressure, and n is the total number of the input features (variables) of a data set.

This feature selection scheme should predefine the number of desired features in the model according to the user's previous knowledge. In practice, however, the optimal size of feature subset is usually unknown, and it should be optimized via an optimization model. To select a varying number of features, we used a discrete binary PSO version described in the above section which was proposed by Kennedy and Eberhart [27] instead of Agrafiotis and Cedeno's [28] approach.

3. A PSO-based feature selection and parameters optimization

3.1. Particle representation

To implement our proposed approach, this research used the RBF kernel function (defined by Eq. (3)) for the SVM classifier because the RBF kernel function can analyze higher dimensional data and requires that only two parameters, C and γ be defined [29,30]. When the RBF kernel is selected, the parameters (C and γ) and features used as input attributes must be optimized using our proposed PSO–SVM system.

The particle therefore, is comprised of three parts, the features mask (discrete-valued), C (continuous-valued), and γ (continuous-valued), when the RBF kernel is selected. Table 1 shows the representation of particle i with dimension of $n_F + 2$, where n_F is the number features that varies from different datasets. The feature mask is Boolean that ''1'' represents the feature is selected, and ''0'' indicates feature is not selected.

3.2. Fitness definition

Classification accuracy and the number of selected features are the two criteria used to design a fitness function. Thus, for the particle with high classification accuracy and a small number of features produce a high fitness value. We solve the multiple criteria problem by creating a single objective fitness function that combines the three goals into one. As defined by formula (11), the fitness has two predefined weights: (i) w_A for the classification accuracy; and (ii) w_F for the selected feature. The weight accuracy can be adjusted to a high value (such as 100%) if accuracy is the most important. The particle with high fitness value has high probability to effect the other particles' positions of the next iteration, so it should be appropriately defined.

$$\mathrm{fitness}_i = w_A \times \mathrm{acc}_i + w_F \times \left[1 - \frac{\left(\sum_{j=1}^{n_F} f_i\right)}{n_F}\right] \quad (11)$$

Table 1
Particle i is comprised of three parts: input feature mask, C, and γ

	Particle type							
	Input feature mask (discrete)						C (continuous)	γ (continuous)
Representation	$x_{i,1}$	$x_{i,2}$	...	$x_{i,d}$	...	x_{i,n_F}	x_{i,n_F+1}	x_{i,n_F+2}

w_A is the weight for the SVM classification accuracy, acc_i the SVM classification accuracy, w_F the weight for the number of selected features, f_j the value of feature mask—"1" represents that feature j is selected and "0" represents that feature j is not selected, and n_F is the total number of features.

For the fitness definition, the classification accuracy (acc) or hit rate denoting the percentage of correctly classified examples is evaluated by Eq. (12). The numbers of correctly and incorrectly classified examples are indicated by cc and uc, respectively.

$$\text{acc} = \frac{\text{cc}}{\text{cc} + \text{uc}} \times 100\% \tag{12}$$

3.3. Strategies for setting the inertia weight and PSO learning process

This study uses constant, decreasing, and random strategies as shown in the following for setting the inertia weight:

(1) Constant: An inertia weight is set to 1.0.
(2) Decreasing: Linearly decreasing inertia weight.

$$w = w\max - \frac{I_{\text{current}}}{I_{\max}}(w\max - w\min) \tag{13}$$

I_{current} is the current iteration, $I_{\max}$ is the predefined maximum iteration and $w\max, w\min$ is the predefined maximum and minimum inertia weight
(3) Random: The inertia weight is defined as follows:

$$w = 0.5 + \left(\frac{\text{rnd}()}{2.0}\right) \tag{14}$$

rnd( ) is the random number in the range [0, 1].

Based on the particle representation, fitness design, and the inertia weight strategy, the procedure for the PSO is as follows.

```
Initialize the population
Do {
    For each particle, do {
        Evaluate current fitness, and set the new pbest
        }
    Choose the particle with the best fitness value among all the particles as the
      gbest.
    For each particle, do {
        Calculate new velocity v_{i,d}(t) by eq. (5) for both continuous and discrete
          type dimensions.
        Update new position x_{i,d}(t) by eq. (7) for continuous type dimensions,
            and by eq. (8) for discrete type dimensions.
        }
} until termination criterion is met
```

4. Numerical illustrations

4.1. Data descriptions

A simulated data set modified from Punch et al. [31] was tested to demonstrate the classificatory accuracy and to realize whether the proposed model can correctly select the relevant features. There are eight target classes that need to be classified in this data set. The data set has 30 features that only five of them (f5, f10, f15, f20, and f25) are relevant to the eight classes. Therefore, there are $2^{30} - 1$ possible feature subsets, and each subset has many possible parameters pairs, C and gamma, to be selected by the search optimizer. Thus, for this demonstrated data set, the feature selection and model parameters are difficult and require huge computation. Table 2 represents a template from which we could generate as many examples as we need. The expression "0.0–1.0" represents uniform random values generated in the ranges of 0.0–1.0. The expression 0.2 + [f10] represents a generation expression where [f10] means that the present value of the f10 field is used in the calculation. Thus features f5, f10, f15, f20, and f25 are fields that can be simultaneously used to distinguish the eight classes A–H, while other 25 features are uniform random noise generated in the range [0.0, 1.0]. Each class comprises 200 cases; therefore, 1600 cases are generated in the data set.

Table 2
A template for generating a data set that consists of 30 features with 8 classes

Class	f5	f10	f15	f20	f25	Others
A	0.0–0.1	0.0–0.1	0.2 + [f10]	$0.5 \times [f10]^2$	$0.5 + 5 \times [f10]^2$	0.0–1.0
B	0.0–0.1	0.0–0.1	0.2 + [f10]	$0.5 \times [f10]^2$	$1.0 + 25 \times [f10]^2$	0.0–1.0
C	0.0–0.1	0.1–0.2	0.3 − [f10]	$0.5 \times [f10]^2$	$0.5 + 5 \times [f10]^2$	0.0–1.0
D	0.0–0.1	0.1–0.2	0.3 − [f10]	$0.5 \times [f10]^2$	$1.0 - 25 \times [f10]^2$	0.0–1.0
E	0.1–0.2	0.0–0.1	0.2 + [f10]	$0.5 \times [f10]^2$	$0.5 + 5 \times [f10]^2$	0.0–1.0
F	0.1–0.2	0.0–0.1	0.2 + [f10]	$0.5 \times [f10]^2$	$1.0 + 25 \times [f10]^2$	0.0–1.0
G	0.1–0.2	0.1–0.2	0.3 − [f10]	$0.5 \times [f10]^2$	$0.5 + 5 \times [f10]^2$	0.0–1.0
H	0.1–0.2	0.1–0.2	0.3 − [f10]	$0.5 \times [f10]^2$	$1.0 - 25 \times [f10]^2$	0.0–1.0

4.2. Experimental procedures for training and testing

To guarantee valid results for making predictions regarding new data, the data set was further randomly partitioned into training sets and independent test sets via a k-fold cross validation. Each of the k subsets acted as an independent holdout test set for the model trained with the remaining $k-1$ subsets. The advantages of cross validation are that all of the test sets were independent and the reliability of the results could be improved. The data set is divided into k subsets for cross validation. A typical experiment uses $k = 10$. Other values may be used according to the data set size. For a small data set, it may be better to set larger k, because this leaves more examples in the training set [32]. This study used $k = 10$, meaning that all of the data will be divided into ten parts, each of which will take turns being the test data set. The other nine data parts serve as the training data set for adjusting the model prediction parameters.

All of the input variables are scaled during the data preprocessing stage. The main advantage of scaling is to avoid attributes in greater numerical ranges dominating those in smaller numerical ranges. Another advantage is to avoid numerical difficulties during the calculation [29]. Feature value scaling can help increase SVM accuracy according to our experimental results. Generally, each feature can be linearly scaled to the [0, 1] range using the following formula (15), where x is original value, x' is scaled value, $\max_a$ is the maximum value of feature a, and $\min_a$ is the minimum value of feature a.

$$x' = \frac{x - \min_a}{\max_a - \min_a} \tag{15}$$

Randomly split the data into ten groups using stratified 10-fold cross validation. Each group contains training, validation and test sets. The training set is used to build the SVM model. The validation set is used to determine the proper training iteration to avoid overtraining. The test set is used to evaluate the model's classification accuracy. For each dataset group, detailed experimental procedure for training and testing is as follows:

Step 1. *Data preparation*: Training, validation, and test sets are represented as Tr, Va, and Te, respectively.
Step 2. *Particle initialization and PSO parameters setting*: Generate initial particles comprised of the feature mask, C, and γ. Set the PSO parameters including number of iterations, velocity limitation, number of particles, particle dimension, and weight for fitness calculation. Set iteration = 0, and perform the training process from step 3–8.
Step 3. Set iteration $i = i + 1$.
Step 4. *SVM model training*:
  (a) Training and validation set preprocessing: select input features for training and validation data sets according to the feature mask which is represented in the first part of a particle.
  (b) SVM classifier accuracy calculation: for the training set Tr, conduct 5-fold cross validation (CV) on the training set, and calculate the average CV accuracy based on the (C, γ) which is represented in the second and third part of a particle.
  (c) Evaluated the classification accuracy on validation set Va using the trained model based on the (C, γ) and the whole training set Tr.
Step 5. *Fitness evaluation*: For each particle, evaluate its fitness by formula (11). Note that the accuracy$_i$ in formula (11) is set as average CV accuracy obtained in the previous step.
Step 6. Update the global and personal best according to the fitness evaluation results. Record the average training CV accuracy and validation accuracy for the global and personal best.
Step 7. *Particle manipulations*: Each particle moves to its next position using formula (7) and (8).
Step 8. *Stop condition checking*: If stopping criteria (maximum iterations predefined) are met, go to step 3, otherwise, go to the next step.
Step 9. To avoid overtraining, we observe the validation accuracy curve, and stop training when the iteration has the best validation accuracy during the training process.
Step 10. With the stopping training iteration determined in the previous step, recall the recorded (in database) feature mask, C, and gamma in the stopping iteration. Retrain and build the SVM classifier on the larger set Tr + Va based on the selected feature subset and SVM model parameters. Finally, measure testing accuracy on test set Te via the trained SVM classifier.
Step 11. End the training and testing procedure.

4.3. System implementation details

The swarm size is set to 60 particles. The searching ranges for C and γ are as follows: $C \in [10^{-2}, 10^{-4}]$ and $\gamma \in [10^{-4}, 10^{3}]$. This study set $v_{\max}$ at about 2–10% of the dynamic range of the variable on each dimension for the continuous type of dimensions, achieving a better and stable convergence in our preliminary experiments. Therefore, $[-v_{\max}, v_{\max}]$ is predefined as [−200, 200] for parameter C, and as [−100, 100] for parameter γ. For the discrete type particle for feature mask, we set $[-v_{\max}, v_{\max}] = [-6, 6]$, which yields a range of [0.9975, 0.0025] using the sigmoid limiting transformation (Eq. (9)). Preliminary experiments also let this study set the personal and social learning factors (c_1, c_2) = (2, 2) that achieves better classification accuracy. The accuracy's weight w_A is adjusted to 95%, and the feature size's weight w_F is set to 5%. These system parameter settings are summarized in Table 3.

Table 3
Setting of the system parameters

	C	γ	Feature mask
Search range	$[10^{-2}, 10^{4}]$	$[10^{-2}, 10^{3}]$	0, 1
Velocity: $[-v_{\max}, v_{\max}]$	[−200, 200]	[−50, 50]	[−6, 6]
Learning factors: (c_1, c_2)	(2, 2)	(2, 2)	(2, 2)

Table 4
The accuracy is illustrated with the form of "average ± standard deviation"

Inertia weight	Accuracy (%)		
	Training set	Validation set	Test set
Decreasing	98.32 ± 2.23	94.93 ± 3.93	95.34 ± 4.67
Constant	97.95 ± 3.23	95.21 ± 4.10	94.36 ± 5.34
Random	97.33 ± 3.83	95.74 ± 4.45	94.23 ± 4.82

Our implementation platform was carried out on the Matlab 7.0, a mathematical development environment, by extending the Libsvm version 2.82 which is originally designed by Chang and Lin [33]. The empirical evaluation was performed on AMD Athlon 64 X2 Duo Core CPU running at 2.01 GHz and 1 G MB RAM.

4.4. Experimental results

A particle contains the feature subset mask, optimal C and gamma. The predefined maximum iteration is 200 in order to avoid overtraining. However, we observe the validation accuracy curve and stop training when the iteration has the best validation accuracy during the training process. When the optimal iteration is determined or the maximum iteration reached, the accuracy of test set is then calculated by the predicted output of the trained SVM classifier. Table 4 shows the average training, validating, and test accuracy on the ten-fold results. Their test accuracies are 95.34 ± 4.67, 94.36 ± 5.34, and 94.23 ± 4.82 respectively, illustrated with the form of "average ± standard deviation."

The three strategies used the same training and test set during each of the ten runs (10-fold cross validation), thus we performed a nonparametric Friedman test—a test for the k-related (dependent) samples, to compare classification accuracies of the test, training, and validation set. We found that for the test, training, and validating accuracy, no significant differences (with significant level of 0.63, 0.55 and 0.43, respectively) among the three inertia weight setting strategies. That means the three inertia weight strategies achieved the same classificatory accuracy, and for this dataset here, the three inertia weight setting strategies can be properly adopted.

Table 5 shows the size of the features, the number of hits on correct features (HIT_F), the number of times the selected feature subset covered the correct features (COVER_F), and the ratio of correct features (RATIO_F) for the ten experiments (10-fold CV). Let f denote the selected feature subset by the PSO, and F denote correct discriminating features (f5, f10, f15, f20, and f25 in this experiment), these three performance indexes are defined as follows:

$$\text{HIT}_F = \frac{\sum_f \text{Hit}_f}{\text{Number of experiment}}; \quad \text{Hit}_f = 1, \text{ if } f = F; \quad \text{Otherwise, } \text{Hit}_f = 0 \tag{16}$$

$$\text{COVER}_F = \frac{\sum_f \text{Cover}_f}{\text{Number of experiment}}; \quad \text{Cover}_f = 1, \text{ if } f \subseteq F; \quad \text{Otherwise, } \text{Cover}_f = 0 \tag{17}$$

$$\text{RATIO}_F = \frac{\text{size of } F}{\text{size of } f} \tag{18}$$

A feature selection system with high HIT_F, COVER_F, and RATIO_F indicates that it successfully chose the correct discriminating features. In this experiment, all of the three strategies successfully selected the correct feature subset with $\text{COVER}_F = 1.0$; that is, their selected feature subsets included all of the correct discriminating features, f5, f10, f15, f20, f25. They also had a large number of correct feature hits (HIT_F) with 0.7, 0.7, and 0.6 respectively. For the correct feature ratio, the random inertia weight strategy (0.82 ± 0.25) is slightly inferior to other two strategies (0.90 ± 0.17 and 0.88 ± 0.20 respectively). A more detail information about the selected features for the decreasing inertia weight strategy is shown in Table 6.

In addition to particle size of 60, we also tried particle sizes of 40 and 100. We found the particle size did not significantly affect the classification accuracy and feature subset performance in this data set. However, it did have an impact on the terminating iterations for searching for the optimal solution with the correct feature subset and classification accuracy. This, thus, shortened or prolonged the execution time. That is, the particle size impacts the convergence. Only the fold #1 experiment with a particle size of 60 is demonstrated here for simplicity. As indicated in Fig. 1, its fitness curves gradually improved from iteration 0 to 200, and exhibited no significant improvements after iteration 150 for the three inertia weight setting strategies. The optimal stopping iteration to get the highest validation accuracy for the three strategies was around iteration 90–130. Based on the best feature mask and SVM parameters (the best particle) for this optimal iteration number, the classification accuracy was calculated according to the training model created by the training set.

Table 5
Summary of feature selection performance index

Inertia weight	Size of features[a]	Correct feature hit: HIT_F	Correct feature coverage: COVER_F	Correct feature ratio*: RATIO_F
Decreasing	5.80 ± 1.40	0.7	1.0	0.90 ± 0.17
Constant	6.10 ± 1.91	0.7	1.0	0.88 ± 0.20
Random	6.80 ± 2.62	0.6	1.0	0.82 ± 0.25

[a] Average ± standard deviation.

Table 6
Detail selected features for decreasing inertia weight strategy

Fold	Features	Size of features	Correct feature	Correct feature coverage	Correct feature ratio
#1	f5, f10, f15, f20, f25	5	Yes	Yes	5/5
#2	f5, f10, f15, f20, f25	5	Yes	Yes	5/6
#3	f5, f10, f12, f15, f20, f25, f27	7	No	Yes	5/7
#4	f5, f10, f15, f20, f25	5	Yes	Yes	5/5
#5	f5, f10, f15, f20, f25	5	Yes	Yes	5/5
#6	f5, f6, f10, f15, f20, f25, f22	7	No	Yes	5/7
#7	f2, f5, f10, f12, f14, f15, f20, f25, f26	9	No	Yes	5/9
#8	f5, f10, f15, f20, f25	5	Yes	Yes	5/5
#9	f5, f10, f15, f20, f25	5	Yes	Yes	5/5
#10	f5, f10, f15, f20, f25	5	Yes	Yes	5/5
	Average	5.80 ± 1.40	0.7	1.0	0.90 ± 0.17

5. Distributed architectures for the PSO–SVM system

5.1. Distributed architectures

The drawback of evolutionary-based systems such as the GA and PSO is the long training time when dealing with a large-scale dataset. It is a well known that many KDD applications require the capability to efficiently process large databases. In such cases, algorithms that offer very good classification accuracy at the cost of high computational complexity cannot be applied. Fortunately, the PSO-based system is well suited for distributed parallel architecture.

Due to the time consuming data mining task, distributed data mining system development is essential. The development of cost-effective web-based interoperable data mining tools must overcome various interoperability issues in heterogeneous environments, namely programming language, platform, object model, and data access [24]. Certain architectures can be adapted to implement distributed systems [34]. In this paper we adopted a web service for our distributed PSO–SVM system, due to its flexible, interoperable, and easy to implement. The experimental results revealed that effective optimization can easily be performed using the web service.

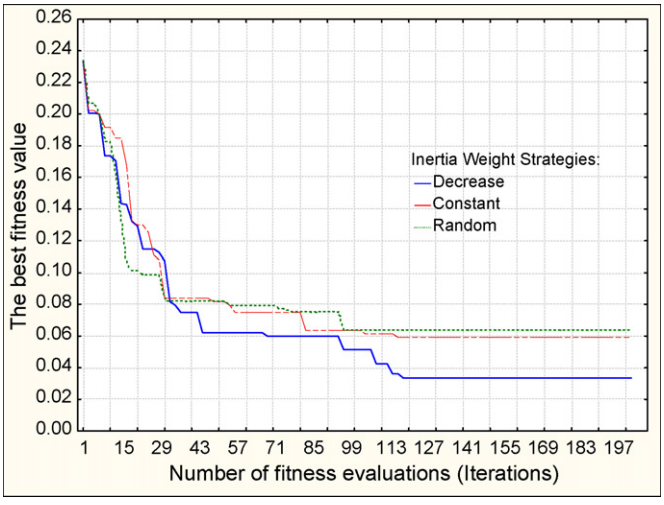

Fig. 1. The fitness value during the training stage for fold #1.

Web services use XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), and UDDI (Universal Description Discovery and Integration) to achieve interoperability in heterogeneous environments. Web service allows applications to easily exchange various data over the Internet as SOAP combines the widespread use of HTTP with the XML ability to represent any data type. Employing XML and HTTP, SOAP is easily utilized in most programming languages.

A three-tier application architecture of the distributed PSO–SVM system is proposed in this study (Fig. 2). Consuming lots of time, the fitness evaluation of each particle that represents a solution is performed in the distributed client site. The particle movement update is performed in the application server site. Restated, the client prepares feature subset, constructs a new SVM classification, calculates classificatory accuracy, and reports the fitness value to the application server. The application server finds the global and personal best of each particle according to particle's fitness, and updates the new position for each particle using the particle's velocity and update formula. The flow of information is as follows: a request comes from the client to the application server; the application server requests/stores data from/to the database server; the database sever returns information to the application server; and the application server returns information back to the client.

(1) In the client, the agents request a task that perform SVM classification based on the particle information which representing the selected features and SVM model parameters—C and gamma. The SVM classifier was modified from the Libsvm developed by Chang and Lin [33].
(2) In the application server, the PSO algorithm was implemented using C# programming language to perform the optimization process that searches the best solution represented by a particle—C, γ, and the feature mask. The task management logic rules designed in the application sever dynamically dispatches requested tasks from all client agents.
(3) In the database server, particle information, fitness value of each particle of each iteration, and related system parameters are stored in the database server.

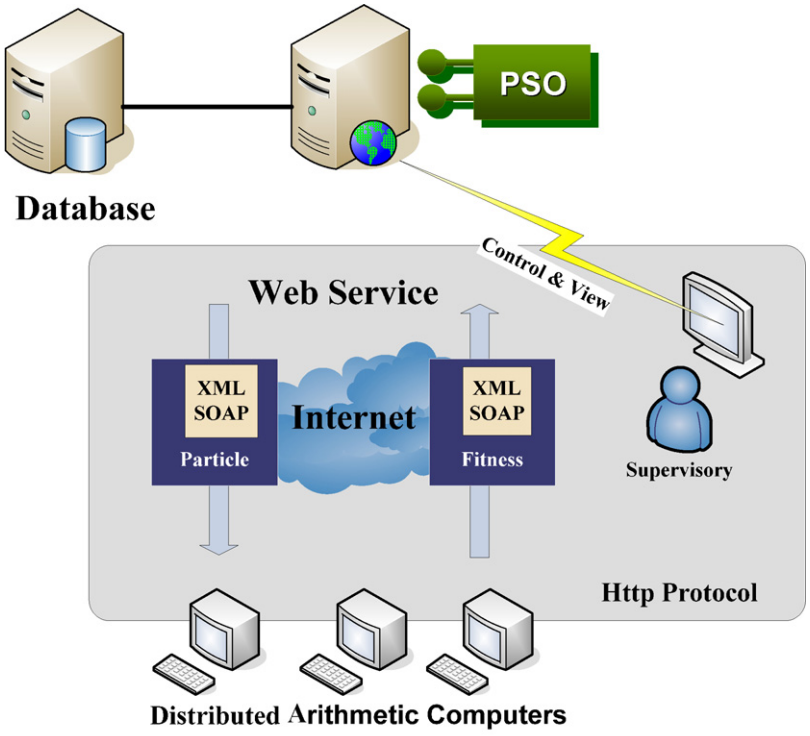

Fig. 2. The three-tier architecture of the distributed PSO–SVM system.

A more detail information flows of the distributed PSO–SVM system are illustrated in Fig. 3. This detail flow is modified and extended from the system procedure described in Section 4.2. However, this flow incorporates the information roles of clients, application server, and database server. Fig. 4 shows the function screen and running process of the SVM classifier in the client agent. The heterogeneous platforms for application server, database server, and client agents are shown

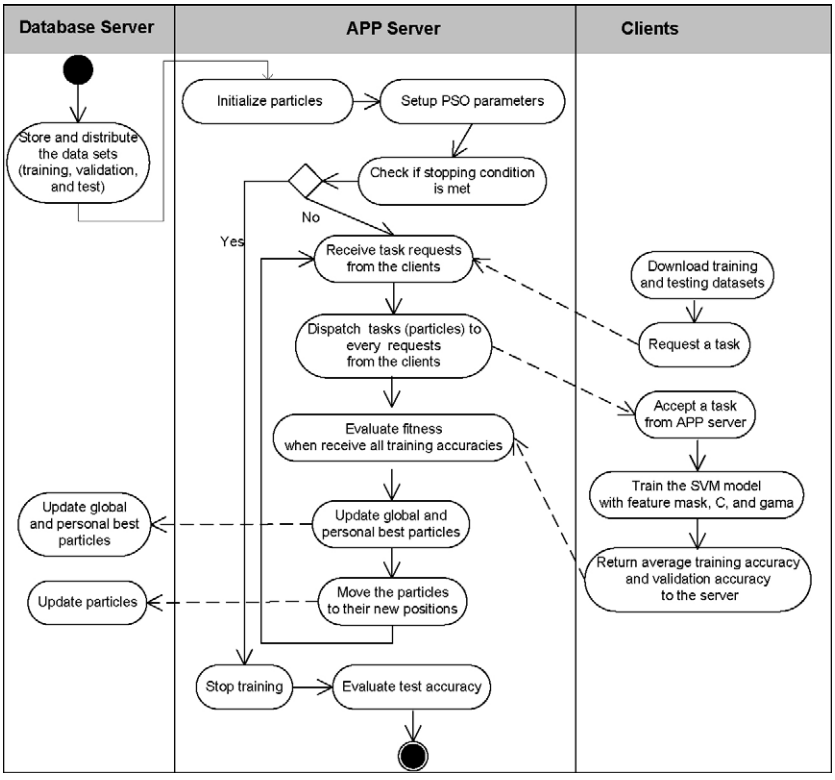

Fig. 3. The system flows of the distributed PSO–SVM system.

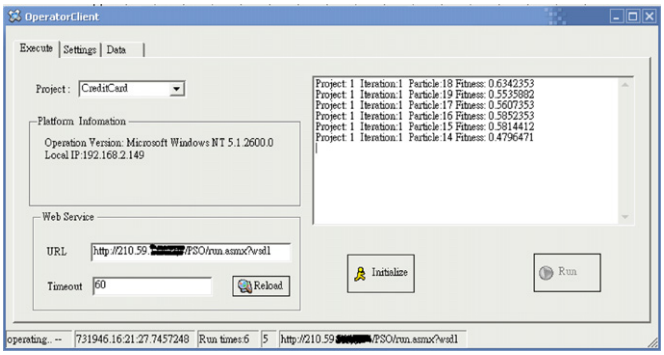

Fig. 4. The function screen and running process of the SVM classifier in the client agent.

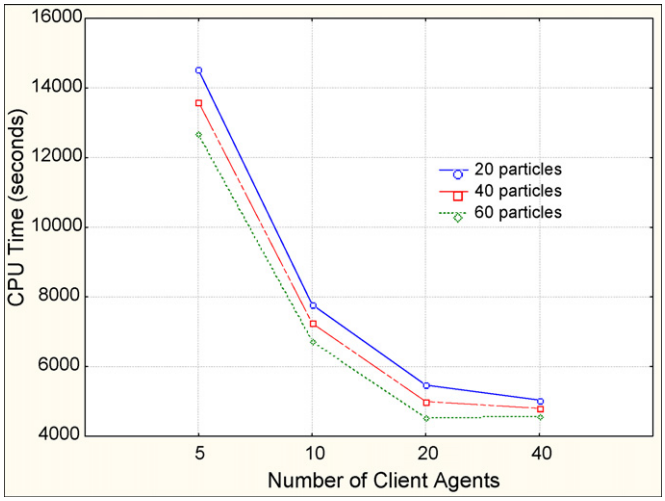

Fig. 5. The CPU times of 5, 10, 20, and 40 clients (with different size of swarm: 20, 40 and 60).

in Table 7. From the heterogeneous platforms used in this study, we can note that the proposed architecture is capable of dealing with unbalance computing capability in a heterogeneous computing environment.

5.2. Evaluation of distributed architecture using a public data set

To demonstrate the proposed distributed hybrid system and evaluate the predictive accuracy, a real world data set, German credit data sets available from the UCI Repository of Machine Learning Databases [35], is adopted in this study. The German credit scoring data consists of 700 instances of creditworthy applicants and 300 instances where credit should not be extended. For each applicant, 24 input variables describe the credit history, account balances, loan purpose, loan amount, employment status, personal information, age, housing, and job title. This data set also consists of a mixture of categorical and continuous attributes.

This study demonstrated the execution time for 5, 10, 20, and 40 client agents with 20 particles (1000 iterations), 40 particles (500 iterations), and 60 particles (250 iterations). We set the stopping iterations opposite of the swarm size because an experiment with small swarm size may need a large number of search iterations to find the optimal. For example, stopping at iteration 1000 for 20 particles is larger than stopping at iteration 500 for 40 particles. Three inertia weight strategies can be applied; for demonstration purpose, however, we used only the constant inertia weight strategy. The system implementation details are the same as those defined in Section 4.3, including personal and social learning factors, ranges of C and gamma, the limits of velocity, and the weights for feature size and classification accuracy. This experiment was conducted with stratified 10-fold cross validation. That is the data set is randomly partitioned into training, validation and test sets via a stratified sampling in which the sample proportion in each data subset is the same as that in the population.

For the classification accuracy and feature selection in the experiment, the test accuracy results for 20, 40 and 60 particles achieved high accuracies. Take the client number of 5 with 20 particles and 1000 iterations for example. The experimental results show that the average accuracy is $77.82 \pm 3.98\%$, and the number of selected features is 16.2 ± 2.75 for 10-fold cross validation experiments. The average size of the selected feature subset is 17.6 ± 1.98; thus the size of the feature subset can be reduced without degrading the accuracy. These results are similar to those found in Ong et al. [36], which reported that the average test accuracies for the GP, BPN and C4.5 are 77.34, 75.51, and 73.17%, respectively, for the German data set. Although our average test accuracy is not as high as 85.6% by the GA-based approach that is reported in Huang and Wang [3], the merit of this study is its validation process to determine a proper training iteration which will not lead to overfitting.

The CPU times can be reduced when the number of client agents is increased, as shown in Fig. 5. However, the execution times did not linearly degrade with the number of client agents, as a bottleneck load appeared in the application server when the amount of communications with the clients increased. The location of the bottleneck depends on the capacities of the application server, database server, the computational loads on clients, and the network load. The distributed PSO–SVM

Table 7
The platform for application server, database server, and three types of client agent

	Application server	Database server	Clients (type I)	Clients (type II)	Clients (type III)
CPU	P3-1 GHz	P4-1.6 GHz	P4-2.0 GHz	P3-1 GHz	P4-1.6 GHz
Memory	128 MB	512 MB	256 MB	256 MB	256 MB
O.S.	Windows 2003 Server	Windows 2003 Server, MS-SQL	Windows XP	Windows 95	Windows 2000

system is flexible in a complex, unbalance, and heterogeneous environment. This distributed architecture is especially essential when the computational load on the client computer is high. When dealing with other domain applications with various dataset sizes, we should balance the load among the application server, database server, clients and the network traffic to reduce the bottleneck load and achieve good computational performance in the entire distributed system.

6. Conclusions

For the data mining system, input feature subset selection and the kernel parameters setting are crucial problems. This study proposed a new hybrid PSO–SVM system to solve these two problems. Two types of PSO versions, namely continuous-valued and discrete version were combined to optimize the best SVM model parameters and the optimal feature subset. The experimental results showed that the proposed system can optimize the model parameters and search the discriminating feature subset simultaneously.

To overcome the long training time when dealing with a large-scale dataset, the PSO–SVM can be implemented with a distributed parallel architecture. This study demonstrated the feasibility of implementing the proposed distributed data mining system via the web service technology. Client computer requests a classificatory task via web service provided in the application server, thus the number of client computers can be scalable and the required hardware specification of client computers can be minimized. This study demonstrated this advantage under a heterogeneous computing environment. However, to reduce the bottleneck load and achieve better computational performances, the distributed system should be properly tuned to balance the load among the application server, database server, clients and the network traffic.

This research is novel, since no empirical research has been carried out on the hybrid PSO–SVM data mining system. It is also a new design to hybridize the continuous and discrete PSO for feature selection and SVM parameter setting. Additionally, the proposed distributed architecture via the web service can be easily implemented in a heterogeneous computing environment.

Acknowledgement

The authors would like to thank the National Science Council of the Republic of China, Taiwan, for financially supporting this research under Contract No. NSC 94-2213-E-327-007.

References

[1] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, NY, USA, 1995.
[2] G.P. Zhang, Neural networks for classification: a survey, IEEE Trans. Syst. Man Cybern.-Part C: Appl. Rev. 30 (4) (2000) 451–462.
[3] C.-L. Huang, C.-J. Wang, A GA-based attribute selection and parameter optimization for support vector machine, Expert Syst. Appl. 31 (2) (2006) 231–240.
[4] H. Fröhlich, O. Chapelle, Feature selection for support vector machines by means of genetic algorithms, in: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, Sacramento, CA, USA, (2003), pp. 142–148.
[5] C.W. Hsu, C.J. Lin, A simple decomposition method for support vector machine, Mach. Learn. 46 (1-3) (2002) 219–314.
[6] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, (1995), pp. 1942–1948.
[7] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2001.
[8] F. van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Inf. Sci. 176 (8) (2006) 937–971.
[9] C.-W. Jiang, B. Etorre, A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimization, Math. Comput. Simul. 68 (1) (2005) 57–65.
[10] F. Du, W. Shi, L. Chen, Y. Deng, Z. Zhu, Infrared image segmentation with 2-D maximum entropy method based on particle swarm optimization, Pattern Recogn. Lett. 26 (5) (2005) 597–603.
[11] P.-Y. Yin, S.-S. Yu, P.-P. Wang, Y.-T. Wang, A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems, Comput. Stand. Interfaces 28 (4) (2006) 441–450.
[12] Y. Da, G. Xiurun, An improved PSO-based ANN with simulated annealing technique, Neurocomputing 63 (2005) 527–533.
[13] A. Chatterjee, P. Siarry, Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, Comput. Operations Res. 33 (3) (2006) 859–871.
[14] X.-H. Shi, Y.-C. Liang, H.-P. Lee, C. Lu, L.-M. Wang, An improved GA and a novel PSO-GA-based hybrid algorithm, Inf. Process. Lett. 93 (5) (2005) 255–261.
[15] A. Allahverdi, F.S. Al-Anzi, A PSO and a tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application, Comput. Operations Res. 33 (4) (2006) 1056–1080.
[16] Z. Lian, X. Gu, B. Jiao, A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan, Appl. Math. Comput. 175 (1) (2006) 773–785.
[17] R. Mendes, J. Kennedy, J. Neves, The informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (3) (2004) 204–210.
[18] S.-T. Li, T.-S. Li, Interoperable web-based data mining system by java distributed object computing, in: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001.
[19] V. Kecman, Learning and Soft Computing, MIT Press, Cambridge, MA, USA, 2001.
[20] B. Schőlkopf, A.J. Smola, Statistical Learning and Kernel Methods, MIT Press, Cambridge, MA, USA, 2000.
[21] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, UK, 2000.
[22] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, V. Vapnik, Comparison of classifier methods: a case study in handwriting digit recognition, in: International Conference on Pattern Recognition, IEEE Computer Society Press, 1994, pp. 77–87.
[23] S. Knerr, L. Personnaz, G. Dreyfus, Single-layer learning revisited: a stepwise procedure for building and training a neural network, in: J. Fogelman (Ed.), Neurocomputing: Algorithms, Architectures and Application, Springer-Verlag, 1990.
[24] U. KreBel, Pairwise Classification and Support Vector Machines, Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, USA, 1999, pp. 254–268.
[25] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, NJ, (1998), pp. 69–73.
[26] R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, application and resources, in: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, 2001, pp. 81–86.
[27] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of the World Multiconference on

Systemics, Cybernetics and Informatics, Piscataway, NJ, (1997), pp. 4104–4109.

[28] D.K. Agrafiotis, W. Cedeno, Feature selection for structure–activity correlation using binary particle swarms, J. Med. Chem. 45 (5) (2002) 1098–1107.

[29] C.W. Hsu, C.C. Chang, C.J. Lin, A practical guide to support vector classification, available at: http://www.csie.ntu.edu.tw/∼cjlin/papers/guide/guide.pdf, 2003.

[30] H.T. Lin, C.J. Lin, A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods, Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, available at: http://www.csie.ntu.edu.tw/∼cjlin/papers/tanh.pdf, 2003.

[31] W.F. Punch, E.D. Goodman, M. Pei, C.-S. Lai, P. Hovland, R. Enbody, Further research on feature selection and classification using genetic algorithms, in: Stephanie Forrest (Ed.), Fifth International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kaufmann, (1993), pp. 557–564.

[32] S.L. Salzberg, On comparing classifiers: pitfalls to avoid and a recommended approach, Data Min. Knowl. Discov. 1 (1997) 317–327.

[33] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, Software available at: http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2001.

[34] V. Gorodetsky, O. Karsaeyv, V. Samoilov, Software tool for agent-based distributed data mining, in: International Conference on Integration of Knowledge Intensive Multi-agent Systems (KIMAS), Cambridge, MA, (2003), pp. 710–715.

[35] P.M. Murphy, D.W. Aha, UCI Repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, CA, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2001.

[36] C.-S. Ong, J.-J. Huang, G.-H. Tzeng, Building credit scoring models using genetic programming, Expert Syst. Appl. 29 (1) (2005) 41–47.