Accelerated Particle Swarm Optimization and Support Vector Machine for Income Prediction and Project Scheduling

Xin-She Yang[1], Suash Deb[2], Simon Fong[3]
[1]Department of Engineering
University of Cambridge
Trumpinton Street
Cambridge CB2 1PZ, UK
[2]Department of Computer Science & Engineering
C. V. Raman College of Engineering
Bidyanagar, Mahura, Janla
Bhubaneswar 752054, India
[3]Department of Computer and Information Science
Faculty of Science and Technology
University of Macau
Taipa, Macau
{xy227@cam.ac.uk, suashdeb@gmail.com, ccfong@umac.mo}

ABSTRACT: Accelerated Particle Swarm Optimization has potential i n many applications and the business sector is one in such wider scope. *Coupled with the support vector machine and metaheuristics the particle swarm optimization is now widely used in solving tough optimization problems. By considering the potential features of these two, we have developed an integrated framework for solving business optimization problems. Basically the proposed APSO-SVM to production optimization is used followed by the income prediction and project scheduling. Finally we tested the framework both by parametric studies as well as discussing the advantages of the proposed metaheuristic SVM.*

1. Introduction

Databases are the central to many information technology applications. This fact is highlighted in the studies that address the data processing and other similar issues. This is partly driven by information technology, especially the Internet, and partly driven by the need to extract meaningful knowledge by data mining. To extract useful information among a huge amount of data requires efficient tools for processing vast data sets. This is equivalent to trying to find an optimal solution to a highly nonlinear problem with multiple, complex constraints, which is a challenging task. Various techniques for such data mining and optimization have been developed over the past few decades. Among these techniques, support vector machine is one of the best techniques for regression, classification and data mining [5, 9, 16, 19, 20, 24].

Besides, the metaheuristic algorithms also become powerful for solving tough nonlinear optimization problems [1, 7]. This is evident through a microscopic analysis of many of the recent studies.

[8, 27, 32]. Modern metaheuristic algorithms have been developed with an aim to carry out global search, typical examples are genetic algorithms [6], particle swarm optimisation (PSO) [7], and Cuckoo Search [29, 30]. The efficiency of metaheuristic algorithms can be attributed to the fact that they imitate the best features in nature, especially the selection of the fittest in biological systems which have evolved by natural selection over millions of years. Since most data have noise or associated randomness, most these algorithms cannot be used directly. In this case, some form of averaging or reformulation of the problem often helps. Even so, most algorithms become difficult to implement for such type of optimization.

In addition to the above challenges, business optimization often concerns with a large amount but often incomplete data, evolving dynamically over time. Certain tasks cannot start before other required tasks are completed, such complex scheduling is often NP-hard and no universally efficient tool exists. Recent trends indicate that metaheuristics can be very promising, in combination with other tools such as neural networks and support vector machines [5, 9, 15, 21].

In this paper, we intend to present a simple framework of business optimization using a combination of support vector machine with accelerated PSO. The paper is organized as follows: We first will briefly review particle swarm optimization and accelerated PSO, and then introduce the basics of support vector machines (SVM). We then use three case studies to test the proposed framework. Finally, we discussion its implications and possible extension for further research.

2. Accelerated Particle Swarm Optimization

2.1 PSO
Particle swarm optimization (PSO) was developed by Kennedy and Eberhart in 1995 [7, 8], based on the swarm behaviour such as fish and bird schooling in nature. Since then, PSO has generated much wider interests, and forms an exciting, ever expanding research subject, called swarm intelligence. PSO has been applied to almost every area in optimization, computational intelligence, and design/scheduling applications. There are at least two dozens of PSO variants, and hybrid algorithms by combining PSO with other existing algorithms are also increasingly popular.

PSO searches the space of an objective function by adjusting the trajectories of individual agents, called particles, as the piecewise paths formed by positional vectors in a quasi-stochastic manner. The movement of a swarming particle consists of two major components: a stochastic component and a deterministic component. Each particle is attracted toward the position of the current global best g^* and its own best location x_i^* in history, while at the same time it has a tendency to move randomly.

Let x_i and v_i be the position vector and velocity for particle i, respectively. The new velocity vector is determined by the following formula

$$v_i^{t+1} = v_i^t + \alpha \in_1 \left[g^* - x_i^t \right] + \beta \in_2 \left[x_i^* - x_i^t \right] \qquad (1)$$

where $\in_1$ and $\in_2$ are two random vectors, and each entry taking the values between 0 and 1. The parameters α and β are the learning parameters or acceleration constants, which can typically be taken as, say, $\alpha \approx \beta \approx 2$.

There are many variants which extend the standard PSO algorithm, and the most noticeable improvement is probably to use an inertia function $\theta(t)$ so that v_i^t is replaced by $\theta(t)v_i^t$

$$v_i^{t+1} = \theta v_i^t + \alpha \in_1 [g^* - x_i^t] + \beta \in_2 [x_i^* - x_i^t] \qquad (2)$$

where $\theta \in (0, 1)$ [2, 3]. In the simplest case, the inertia function can be taken as a constant, typically $\theta \approx 0.5 \sim 0.9$. This is equivalent to introducing a virtual mass to stabilize the motion of the particles, and thus the algorithm is expected to converge more quickly.

2.2 Accelerated PSO
The standard particle swarm optimization uses both the current global best g^* and the individual best x_i^*. The reason of using the individual best is primarily to increase the diversity in the quality solutions, however, this diversity can be simulated using some randomness. Subsequently, there is no compelling reason for using the individual best, unless the optimization problem of interest is highly nonlinear and multimodal.

A simplified version which could accelerate the convergence of the algorithm is to use the global best only. Thus, in the accelerated particle swarm optimization (APSO) [27, 32], the velocity vector is generated by a simpler formula

$$v_i^{t+1} = v_i^t + \alpha \in_n + \beta \left[g^* - x_i^t \right] \qquad (3)$$

where $\in_n$ is drawn from $N(0,1)$ to replace the second term. The update of the position is simply

$$x_i^{t+1} = x_i^t + v_i^{t+1} \qquad (4)$$

In order to increase the convergence even further, we can also write the update of the location in a single step

$$x_i^{t+1} = (1-\beta)x_i^t + \beta g^* + \alpha \in_n \qquad (5)$$

This simpler version will give the same order of convergence. Typically, $\alpha = 0.1L \sim 0.5L$ where L is the scale of each variable, while $\beta = 0.1 \sim 0.7$ is sufficient for most applications. It is worth pointing out that velocity does not appear in equation (5), and there is no need to deal with initialization of velocity vectors. Therefore, APSO is much simpler. Comparing with many PSO variants, APSO uses only two parameters, and the mechanism is simple to understand.

A further improvement to the accelerated PSO is to reduce the randomness as iterations proceed. This means that we can use a monotonically decreasing function such as

$$\alpha = \alpha_0 e^{-\gamma t} \qquad (6)$$

or

$$\alpha = \alpha_0 \gamma^t, \qquad (0 < \gamma < 1), \qquad (7)$$

where $\alpha_0 \approx 0.5 \sim 1$ is the initial value of the randomness parameter. Here t is the number of iterations or time steps. $0 < \gamma < 1$ is a control parameter [32]. For example, in our implementation, we will use

$$\alpha = 0.7^t, \qquad (8)$$

where $t \in [0, t_{max}]$ and t_{max} is the maximum of iterations.

3. Support Vector Machine

Support vector machine (SVM) is an efficient tool for data mining and classification [25, 26]. Due to the vast volumes of data in business, especially e-commerce, efficient use of data mining techniques becomes a necessity. In fact, SVM can also be considered as an optimization tool, as its objective is to maximize the separation margins between data sets. The proper combination of SVM with metaheuristics could be advantageous.

3.1 Support Vector Machine
A support vector machine essentially transforms a set of data into a significantly higher-dimensional space by nonlinear transformations so that regression and data fitting can be carried out in this high-dimensional space. This methodology can be used for data classification, pattern recognition, and regression, and its theory was based on statistical machine learning theory [21, 24, 25].

For classifications with the learning examples or data (x_i, y_i) where $i = 1, 2, ..., n$ and $y_i \in \{1, +1\}$, the aim of the learning is to find a function $\phi_\alpha(x)$ from allowable functions $\{\phi_\alpha : \alpha \in \Omega\}$ such that $\phi_\alpha(x_i) \mapsto y_i$ for $(i = 1, 2, ..., n)$ and that the expected risk $E(\alpha)$ is minimal. That is the minimization of the risk

$$E(\alpha) = \frac{1}{2} \int |\phi_\alpha(x) - y| dQ(x, y) \qquad (9)$$

where $Q(x, y)$ is an unknown probability distribution, which makes it impossible to calculate $E(\alpha)$ directly. A simple approach is to use the so-called empirical risk

$$E_p(\alpha) \approx \frac{1}{2n} \sum_{i=1}^{n} |\phi_\alpha(x_i) - y_i| \qquad (10)$$

However, the main flaw of this approach is that a small risk or error on the training set does not necessarily guarantee a small error on prediction if the number n of training data is small [26].

For a given probability of at least 1-p, the Vapnik bound for the errors can be written as

$$E(\alpha) \le R_p(\alpha) + \psi\left(\frac{h}{n}, \frac{log(p)}{n}\right), \tag{11}$$

$$\psi\left(\frac{h}{n}, \frac{log(p)}{n}\right) = \sqrt{\frac{1}{n}\left[h\left(log\frac{2n}{h}+1\right)-log\left(\frac{p}{4}\right)\right]} \tag{12}$$

Here h is a parameter, often referred to as the Vapnik-Chervonenskis dimension or simply VC-dimension [24], which describes the capacity for prediction of the function set ϕ_α.

In essence, a linear support vector machine is to construct two hyperplanes as far away as possible and no samples should be between these two planes. Mathematically, this is equivalent to two equations

$$w \cdot x + b = \pm 1, \tag{13}$$

and a main objective of constructing these two hyperplanes is to maximize the distance (between the two planes)

$$d = \frac{2}{\|w\|} \tag{14}$$

Such maximization of d is equivalent to the minimization of $\|w\|$ or more conveniently $\|w\|^2$. From the optimization point of view, the maximization of margins can be written as

$$\text{minimize } \frac{1}{2}\|w\|^2 = \frac{1}{2}(w \cdot w). \tag{15}$$

This essentially becomes an optimization problem

$$\text{minimize} \, \psi = \frac{1}{2}\|w\|^2 + \lambda\sum_{i=1}^{n}\eta_i \tag{16}$$

$$\text{subject to } y_i(w \cdot x_i + b) \ge 1 - \eta_i, \tag{17}$$

$$\eta_i \ge 0, \qquad (i = 1, 2, ..., n), \tag{18}$$

where $\lambda > 0$ is a parameter to be chosen appropriately. Here, the term $\sum_{i=1}^{n}\eta_i$ is essentially a measure of the upper bound of the number of misclassifications on the training data.

3.2 Nonlinear SVM and Kernel Tricks
The so-called kernel trick is an important technique, transforming data dimensions while simplifying computation. By using Lagrange multipliers $\alpha_i \ge 0$, we can rewrite the above constrained optimization into an unconstrained version, and we have

$$L = \frac{1}{2}\|w\|^2 + \lambda\sum_{i=1}^{n}\eta_i - \sum_{i=1}^{n}\alpha_i[y_i(w \cdot x_i + b) \ge (1-\eta_i)] \tag{19}$$

From this, we can write the Karush-Kuhn-Tucker conditions

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0, \tag{20}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n}\alpha_i y_i = 0, \tag{21}$$

$$y_i(w \cdot x_i + b) - (1-\eta_i) \ge 0, \tag{22}$$

$$\alpha_i[y_i(w \cdot x_i + b) - (1-\eta_i)] = 0, \qquad (i=1,2,...,n), \tag{23}$$

$$\alpha_i \geq 0, \qquad \eta_i \geq 0, \qquad (i = 1, 2,..., n). \qquad (24)$$

From the first KKT condition, we get

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i. \qquad (25)$$

It is worth pointing out here that only the nonzero α_i contribute to overall solution. This comes from the KKT condition (23), which implies that when $\alpha_i \neq 0$, the inequality (17) must be satisfied exactly, while $\alpha_0 = 0$ means the inequality is automatically met. In this latter case $\eta_i = 0$. Therefore, only the corresponding training data (x_i, y_i) with $\alpha_i > 0$ can contribute to the solution, and thus such x_i form the support vectors (hence, the name support vector machine). All the other data with $\alpha_i = 0$ become irrelevant.

It has been shown that the solution for α_i can be found by solving the following quadratic programming [24, 26]

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i . x_j), \qquad (26)$$

subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \qquad 0 \leq \alpha_i \leq \lambda, \qquad (i = 1,2,.....,n). \qquad (27)$$

From the coefficients α_i, we can write the final classification or decision function as

$$f(x) = \text{sgn} [\sum_{i=1}^{n} \alpha_i y_i (x_i . x_j) + b], \qquad (28)$$

where sgn is the classic sign function.

As most problems are nonlinear in business applications, and the above linear SVM cannot be used. Ideally, we should find some nonlinear transformation ϕ so that the data can be mapped onto a high-dimensional space where the classification becomes linear. The transformation should be chosen in a certain way so that their dot product leads to a kernel-style function $K(x, x_i) = \phi(x). \phi(x_i)$. In fact, we do not need to know such transformations, we can directly use the kernel functions $K(x, x_i)$ to complete this task. This is the so-called kernel function trick. Now the main task is to chose a suitable kernel function for a given, specific problem.

For most problems in nonlinear support vector machines, we can use $K(x, x_i) = (x . x_i)^d$ for polynomial classifiers, $K(x, x_i) = \tanh[k(x . x_i) + \Theta)]$ for neural networks, and by far the most widely used kernel is the Gaussian radial basis function (RBF)

$$K(x, x_i) = \exp [- \frac{|| X - X_i ||^2}{(2\sigma^2)}] = \exp [-\gamma || X - X_i ||^2], \qquad (29)$$

for the nonlinear classifiers. This kernel can easily be extended to any high dimensions. Here σ^2 is the variance and $\gamma = 1/2\sigma^2$ is a constant. In general, a simple bound of $0 \leq \gamma \leq C$ is used, and here C is a constant.

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \alpha_i \alpha_j y_i y_j K(x_i . x_j) . \qquad (30)$$

It is worth pointing out under Mercer's conditions for the kernel function, the matrix $A = y_i y_j K(x_i, x_j)$ is a symmetric positive definite matrix [26], which implies that the above maximization is a quadratic programming problem, and can thus be solved efficiently by standard QP techniques [21].

4. Metaheuristic Support Vector Machine with APSO

4.1 Metaheuristics
There are many metaheuristic algorithms for optimization and most these algorithms are inspired by nature [27].Metaheuristic

algorithms such as genetic algorithms and simulated annealing are widely used, almost routinely, in many applications, while relatively new algorithms such as particle swarm optimization [7], firefly algorithm and cuckoo search are becoming more and more popular [27, 32]. Hybridization of these algorithms with existing algorithms are also emerging.

The advantage of such a combination is to use a balanced tradeoff between global search which is often slow and a fast local search. Such a balance is important, as highlighted by the analysis by Blum and Roli [1]. Another advantage of this method is that we can use any algorithms we like at different stages of the search or even at different stage of iterations. This makes it easy to combine the advantages of various algorithms so as to produce better results.

Others have attempted to carry out parameter optimization associated with neural networks and SVM. For example, Liu et al. have used SVM optimized by PSO for tax forecasting [13]. Lu et al. proposed a model for finding optimal parameters in SVM by PSO optimization [14]. However, here we intend to propose a generic framework for combining efficient APSO with SVM, which can be extended to other algorithms such as firefly algorithm [28, 31].

4.2 APSO-SVM

Support vector machine has a major advantage, that is, it is less likely to overfit,compared with other methods such as regression and neural networks. In addition, efficient quadratic programming can be used for training support vector machines. However, when there is noise in the data, such algorithms are not quite suitable. In this case, the learning or training to estimate the parameters in the SVM becomes difficult or inefficient.

Another issue is that the choice of the values of kernel parameters C and σ^2 in the kernel functions; however, there is no agreed guideline on how to choose them, though the choice of their values should make the SVM as efficiently as possible. This itself is essentially an optimization problem.

Taking this idea further, we first use an educated guess set of values and use the metaheuristic algorithms such as accelerated PSO or cuckoo search to find the best kernel parameters such as C and σ^2 [27, 29]. Then, we used these parameters to construct the support vector machines which are then used for solving the problem of interest. During the iterations and optimization, we can also modify kernel parameters and evolve the SVM accordingly. This framework can be called a metaheuristic support vector machine. Schematically, this Accelerated PSO-SVM can be represented as shown in Figure 1.

```
begin
Define the objective;
Choose kernel functions;
Initialize various parameters;
while (criterion)
    Find optimal kernel parameters by APSO;
    Construct the support vector machine;
    Search for the optimal solution by APSO-SVM;
    Increase the iteration counter;
end
Post-processing the results;
end
```

Figure 1. Metaheuristic APSO-SVM

For the optimization of parameters and business applications discussed below, APSO is used for both local and global search [27, 32].

5. Business Optimization Benchmarks

Using the framework discussed earlier, we can easily implement it in any programming language, though we have implemented using Matlab. We have validated our implementation using the standard test functions, which confirms the correctness of the implementation. Now we apply it to carry out case studies with known analytical solution which can be used for comparison, while the second case is a standard benchmark in resource-constrained project scheduling [11].

5.1 Production Optimization

Let us first use the proposed approach to study the classical Cobb-Douglas production optimization. For a production of a series of products and the labour costs, the utility function can be written

$$q = \prod_{j=1}^{n} u_j^{\alpha_j} = u_1^{\alpha_1} u_2^{\alpha_2} ... u_n^{\alpha_n}, \tag{31}$$

where all exponents α_j are non-negative, satisfying

$$\sum_{j=1}^{n} \alpha_j = 1. \tag{32}$$

The optimization is the minimization of the utility

$$\text{minimize } q \tag{33}$$

$$\text{subject to} \sum_{j=1}^{n} w_j u_j = K, \tag{34}$$

where $w_j (j = 1, 2, ..., n)$ are known weights.

This problem can be solved using the Lagrange multiplier method as an unconstrained problem

$$\psi = \prod_{j=1}^{n} u_j^{\alpha_j} + \lambda \left(\sum_{j=1}^{n} w_j u_j - K \right), \tag{35}$$

whose optimality conditions are

$$\frac{\partial \psi}{\partial u_j} = \alpha_j u_j^{-1} \prod_{j=1}^{n} u_j^{\alpha_j} + \lambda w_j = 0, \quad (j = 1, 2, ..., n), \tag{36}$$

$$\frac{\partial \psi}{\partial \lambda} = \sum_{j=1}^{n} w_j u_j - K = 0 \tag{37}$$

The solutions are

$$u_1 = \frac{K}{w_1 [1 + \frac{1}{\alpha_1} \sum_{j=2}^{n} \alpha_1]}, \; u_j = \frac{w_1 \alpha_j}{w_j \alpha_j} u_1, \tag{38}$$

where $(j = 2, 3, ..., n)$. For example, in a special case of $n = 2$, $\alpha_1 = 2/3$, $\alpha_2 = 1/3$, $w_1 = 5$, $w_2 = 2$ and $K = 300$, we have

$$u_1 = \frac{Q}{w_1 (1 + \alpha_2 / \alpha_1)} = 40, \qquad u_2 = \frac{K \alpha_2}{w_2 \alpha_1 (1 + \alpha_2 / \alpha_1)} = 50,$$

As most real-world problem has some uncertainty, we can now add some noise to the above problem. For simplicity, we just modify the constraint as

$$\sum_{j=1}^{n} w_j u_j = K (1 + \beta \in), \tag{39}$$

where $\in$ is a random number drawn from a Gaussian distribution with a zero mean and a unity variance, and $0 \leq \beta \ll 1$ is a small positive number.

We now solve this problem as an optimization problem by the proposed APSO-SVM. In the case of $\beta = 0{:}01$, the results have been summarized in Table 1 where the values are provided with different problem size n with different num- bers of iterations. We can see that the results converge at the optimal solution very quickly.

6. Income Prediction

Studies to improve the accuracy of classifications are extensive. For example, Kohavi proposed a decision-tree hybrid in 1996

size n	Iterations	deviations
10	1000	0.014
20	5000	0.037
50	5000	0.040
50	15000	0.009

Table 1. Mean deviations from the optimal solutions

[10]. Furthermore, an efficient training algorithm for support vector machines was proposed by Platt in 1998 [17, 18], and it has some significant impact on machine learning, regression and data mining.

A well-known benchmark for classification and regression is the income prediction using the data sets from a selected 14 attributes of a household from a sensus form [10, 17].We use the same data sets at ftp://ftp.ics.uci.edu/pub/machine-learning-databases/adult for this case study. There are 32561 samples in the training set with 16281 for testing. The aim is to predict if an individual's income is above or below 50K ?

Among the 14 attributes, a subset can be selected, and a subset such as age,education level, occupation, gender and working hours are commonly used.

Using the proposed APSO-SVM and choosing the limit value of C as 1.25,the best error of 17.23% is obtained (see Table 2), which is comparable with most accurate predictions reported in [10, 17].

Train set (size)	Prediction set	Errors (%)
512	256	24·9
1024	256	20·4
16400	8200	17.23

Table 2. Income prediction using APSO-SVM

6.1 Project Scheduling

Scheduling is an important class of discrete optimization with a wider range of applications in business intelligence. For resource-constrained project scheduling problems, there exists a standard benchmark library by Kolisch and Sprecher [11,12]. The basic model consists of J activities/tasks, and some activities cannot start before all its predecessors h are completed. In addition, each activity $j = 1,2,...,J$ can be carried out, without interruption, in one of the M_j modes, and performing any activity j in any chosen mode m takes d_{jm} periods, which is supported by a set of renewable resource R and non-renewable resources N. The project's makespan or upper bound is T, and the overall capacity of non- renewable resources is K_r^{υ} where $r \in N$. For an activity j scheduled in mode m, it uses k_{jmr}^{ρ} units of renewable resources and k_{jmr}^{υ} units of non-renewable resources in period $t = 1,2,...,T$.

For activity j, the shortest duration is fit into the time windows $[EF_j, LF_j]$ where EF_j is the earliest finish times, and LF_j is the latest finish times. Mathematically, this model can be written as [11].

$$\text{Minimize } \Psi(x) \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} t \cdot x_{jmt} \tag{40}$$

subject to

$$\sum_{m=1}^{M_h} \sum_{t=EF_j}^{LF_j} t \; x_{hmt} \leq \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t - d_{jm}) \; x_{jmt}, \; (j = 2,...,J)$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr}^{\rho} \sum_{q=max\{t,EF_j\}}^{min\{t+d_{jm}-1, LF_j\}} x_{jmq} \leq k_r^{\rho}(r \in R) \tag{41}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr}^{\upsilon} \sum_{t=EF_j}^{LF_j} x_{jmq} \leq k_r^{\upsilon}, (r \in N) \qquad (41)$$

and

$$\sum_{j=1}^{M_j} \sum t=EF_j^{LF_j} =1, \qquad j=1,2,...,j, \qquad (42)$$

Number of iterations	SVM kernel parameters
1000	$C = 149.2$, $\sigma^2 = 67:9$
5000	$C = 127.9$, $\sigma^2 = 64:0$

Table 3. Kernel parameters used in SVM

where $x_{jmt} \in \{0,1\}$ and $t = 1,...,T$. As x_{jmt} only takes two values 0 or 1, this problem can be considered as a classification problem, and metaheuristic support vector machine can be applied naturally.

Using the online benchmark library [12], we have solved this type of problem with $J = 30$ activities (the standard test set j30). The run time on a modern desktop computer is about 2.2 seconds for $N = 1000$ iterations to 15.4 seconds for $N = 5000$ iterations. We have run the simulations for 50 times so as to obtain meaningful statistics.

The optimal kernel parameters found for the support vector machines are listed in Table 3, while the deviations from the known best solution are given in Table 4 where the results by other methods are also compared.

Algorithm	Authors	$N = 1000$	5000
PSO [22]	Kemmoe et al. (2007)	0.26	0.21
hybribd GA [23]	Valls eta al. (2007)	0.27	0.06
Tabu search [15]	Nonobe & Ibaraki (2002)	0.46	0.16
Adapting GA [4]	Hartmann (2002)	0.38	0.22
Meta APSO-SVM	this paper	0.19	0.025

Table 4. Mean deviations from the optimal solution (J=30)

From these tables, we can see that the proposed metaheuristic support vector machine starts very well, and results are comparable with those by other methods such as hybrid genetic algorithm. In addition, it converges more quickly, as the number of iterations increases. With the same amount of function evaluations involved, much better results are obtained, which implies that APSO is very efficient, and subsequently the APSO-SVM is also efficient in this context. In addition, this also suggests that this proposed framework is appropriate for automatically choosing the right parameters for SVM and solving nonlinear optimization problems.

7. Conclusions

Both PSO and support vector machines are now widely used as optimization techniques in business intelligence. They can also be used for data mining to extract useful information efficiently. SVM can also be considered as an optimization technique in many applications including business optimization. When there is noise in data, some averaging or reformulation may lead to better performance. In addition, metaheuristic algorithms can be used to find the optimal kernel parameters for a support vector machine and also to search for the optimal solutions. We have used three very different case studies to demonstrate such a metaheuristic SVM framework works.

Automatic parameter tuning and efficiency improvement will be an important topic for further research. It can be expected that this framework can be used for other applications. Furthermore, APSO can also be used to combine with other algorithms such as neutral networks to produce more efficient algorithms [13, 14]. More studies in this area are highly needed.

References

[1] Blum, C., Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptural comparision, *ACM Comput. Surv.*, 35, 268-308.

[2] Chatterjee, A., and Siarry, P. (2006) Nonlinear inertia variation for dynamic adaptation in particle swarm optimization, *Comp. Oper. Research*, 33, 859-871.

[3] Clerc, M., Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space, *IEEE Trans. Evolutionary Computation*, 6, 58-73.

[4] Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints, *Naval Res. Log.*, 49, 433-448 .

[5] Howley ,T., Madden, M, G.(2005). The genetic kernel support vector machine: description and evaluation, *Artificial Intelligence Review*, 24, 379-395.

[6] Goldberg, D. E.(1989).Genetic Algorithms in Search, Optimisation and Machine Learning, Reading, Mass.: Addison Wesley.

[7] Kennedy, J., Eberhart, R, C. (1995). Particle swarm optimization, *In*: Proc. of IEEE In ternational Conference on Neural Networks, Piscataway, NJ. p. 1942-1948.

[8] Kennedy, J., Eberhart, R, C. (2001). *Swarm intelligence*, Academic Press.

[9] Kim, K. (2003). Financial forecasting using support vector machines, *Neurocomputing*, 55,307-319.

[10] Kohavi, R. (1996). Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid, *In*: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, p. 202-207, AAAI Press. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/adult.

[11] Kolisch, R., Sprecher, A.,(1996). PSPLIB -a project scdeluing problem library, OR Software-ORSEP (operations research software exchange prorgam) by H. W. Hamacher, *Euro. J. Oper. Res.*, 96, 205-216.

[12] Kolisch, R., Sprecher, A. (2011). The Library PSBLIB, http://129.187.106.231/psplib/ 13. Liu L.-X., Zhuang Y. and Liu X. Y., Tax forecasting theory and model based on SVM optimized by PSO, *Expert Systems with Applications*, 38, January 2011, p.116-120

[14] Lu, N., Zhou, J. Z., He, Y., Liu, Y. (2009). Particle Swarm Optimization for Parameter Optimization of Support Vector Machine Model, *In*: 2009 Second International Conference on Intelligent Computation Technology and Automation, IEEE publications, p. 283-284.

[15] Nonobe, K., Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP), *In*: Essays and Surveys in Metaheuristics (Eds. Ribeiro C. C. and Hansen P.), p. 557-588.

[16] Pai, P, F., Hong, W, C. (2005). Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms, *Electric Power Sys. Res.*, 74, 417-425.

[17] Platt J. C., Sequential minimal optimization: a fast algorithm for training support vector machines, Techical report MSR-TR-98014, Microsoft Research, (1998).

[18] Plate, J, C., (1999). Fast training of support vector machines using sequential minimal optimization, *In*: *Advances in Kernel Methods Support Vector Learning* (Eds. B. Scholkopf, C. J. Burges and A. J. Smola), MIT Press, p. 185-208.

[19] Shi, G, R. (2009). The use of support vector machine for oil and gas identification in low-porosity and low-permeability reservoirs, *Int. J. Mathematical Modelling and Numerical Optimisation*, 1, 75-87 .

[20] Shi, G, R., Yang, X, S. (2010). Optimization and data mining for fracture prediction in geosciences, *Procedia Computer Science*, 1, 1353-1360 .

[21] Smola, A, J., SchÄolkopf, B. (1998). A tutorial on support vector regression, http://www.svms.org/regression/

[22] Tchomte, S, K., Gourgand, M., Quilliot, A.(2007). Solving resource-constrained project scheduling problem with particle swarm optimization, *In*: Proceeding of 3rd Multidsciplinary Int. Scheduling Conference (MISTA 2007), 28 - 31 Aug 2007, Paris, p. 251-258.

[23] Valls, V., Ballestin, F., Quintanilla, S.(2007). A hybrid genetic algorithm for the resource-constrained project scheduling problem, *Euro. J. Oper. Res.*.

[24] Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data* (in Russian), Moscow, 1979. English translation published by Springer-Verlag, New York.

[25] Vapnik, V. (1995). *The nature of Statistical Learning Theory*, Springer-Verlag, New York.

[26] Scholkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi P., Poggio, T., and Vapnik, V. (1997). Comparing support vector machine with Gaussian kernels to radial basis function classifiers, *IEEE Trans. Signal Processing*, 45, 2758-2765.

[27] Yang, X, S.(2008). *Nature-Inspired Metaheuristic Algorithms*, Luniver Press.

[28] Yang, X, S.(2009). Firefly algorithms for multimodal optimization, *In*: *Stochastic Algorithms: Foundations and Applications*, SAGA 2009, Lecture Notes in Computer Sciences, 5792, p. 169-178.

[29] Yang, X.S., Deb, S. (2009). Cuckoo search via Levy flights, in: *Proceeings of World Congress on Nature & Biologically Inspired Computing* (NaBIC 2009, India), IEEE Publications, USA, p. 210-214.

[30] Yang, X, S., Deb, S. (2010). Engineering optimization by cuckoo search, *Int. J. Mathematical Modelling and Numerical Optimisation*, 1, 330-343.

[31] Yang, X, S.(2010). Firefly algorithm, stochastic test functions and design optimisation, *Int. J. Bio-inspired Computation*, 2, 78-84 .

[32] Yang X. S., X, S.(2010). *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley & Sons.