EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Dynamic artificial bee colony algorithm for multi-parameters optimization of support vector machine-based soft-margin classifier

Yiming Yan[1*], Ye Zhang[2] and Fengjiao Gao[3]

Abstract

This article proposes a 'dynamic' artificial bee colony (D-ABC) algorithm for solving optimizing problems. It overcomes the poor performance of artificial bee colony (ABC) algorithm, when applied to multi-parameters optimization. A dynamic 'activity' factor is introduced to D-ABC algorithm to speed up convergence and improve the quality of solution. This D-ABC algorithm is employed for multi-parameters optimization of support vector machine (SVM)-based soft-margin classifier. Parameter optimization is significant to improve classification performance of SVM-based classifier. Classification accuracy is defined as the objection function, and the many parameters, including 'kernel parameter', 'cost factor', etc., form a solution vector to be optimized. Experiments demonstrate that D-ABC algorithm has better performance than traditional methods for this optimizing problem, and better parameters of SVM are obtained which lead to higher classification accuracy.

Keywords: Dynamic artificial bee colony algorithm, Multi-parameters optimization, Support vector machine, Soft-margin classifier

Introduction

Artificial bee colony (ABC) algorithm was first proposed by Karaboga in 2005 [1]. It has many advantages than earlier swarm intelligence algorithms, especially for constrained optimization problem.

A constrained optimization problem (1) is defined as finding solution $\vec{x}$ that minimizes an objective function $f(\vec{x})$ subject to inequality and/or equality constraints [2]:

$$\begin{aligned} &\text{minimize } f(\vec{x}), \quad \vec{x} = (x_1, \ldots, x_D) \in \Re^D \\ &\qquad\qquad l_i \le x_i \le u_i, \qquad i = 1, \ldots, D \\ &\text{subject to :} \quad g_j(\vec{x}) \le 0, \qquad for\, j = 1, \ldots, q \\ &\qquad\qquad h_j(\vec{x}) = 0, \qquad for\, j = q+1, \ldots, m \end{aligned} \tag{1}$$

when D is larger and each element of $\vec{x}$ represents a specific parameter, it is a multi-parameters optimization problem.

Simulating the foraging behavior of honey bee swarm, ABC algorithm assumes solution $\vec{x}$ as coordinate of nectar source in D-dimensional space, and defines objective function $f(\vec{x})$ which reflects quality of the nectar source. Small value of objective function indicates better nectar source. As bee swarm continually searching better nectar source, the algorithm could find the best solution $\vec{x}$.

However, ABC algorithm is criticized owing to its poor convergence rate and local optimization problems [3-6]. Many modified methods have been proposed. As the earlier idea of many researchers, poor performance is attributed to 'roulette wheel' selection mechanism, which is introduced in the onlooker phase of the original ABC algorithm. Boltzmann selection mechanism was employed instead of roulette wheel selection by Haijun and Qingxian [7]. Interactive ABC, proposed by Tsai et al. [8], introduced the Newtonian law of universal gravitation, which was also for modifying the original selection mechanism. Akbari et al. [9] proposed a modified formula for different phases of ABC algorithm. Actually, according to testing by abundant experiments, these modified methods could improve the original algorithm

* Correspondence: dragon_no.2@hotmail.com
[1]Department of information engineering, Harbin institute of technology, (92 West Dazhi Street), Harbin (150001), China
Full list of author information is available at the end of the article

only when D is not too large. Nevertheless, our findings provide evidence that it is the 'randomly single element modification (RSEM) process', which principally leads ABC algorithm to poor performance. In traditional ABC algorithm, for each memorized solution $\vec{x}$, modifying operation is on single element x_k ($k \in [1,D]$) of $\vec{x}$ in each cycle, and solution $\vec{x}$ changes little after each modification. Moreover, element x_k is randomly selected. It is uncertain whether the modification of x_k could improve the solution $\vec{x}$, particularly when D is large. Consequently, more cycles are needed for searching best solution, and the algorithm performs poor efficiency relatively. Although Karaboga and Akay [2] introduced modification rate (MR) factor to randomly modify more elements of the solution vector in each cycle, robustness of the algorithm is not quite well. Furthermore, in ABC algorithm, optimization is hierarchical (from global to local), which is implemented mainly by operations of employed bees and onlooker bees, respectively. However, RSEM process is simultaneously utilized in these two phases, which could not effectively guarantee hierarchical optimization. Therefore, RSEM process is abandoned in our D-ABC algorithm. A dynamic 'activity' factor is introduced to modify appropriate number of elements of solution $\vec{x}$ and achieve hierarchical optimization. In different optimizing stages, active degree of bees is properly set. More active bees modify more elements of $\vec{x}$. For bees with different division of labor, 'activity' factors are different set. Thus, hierarchical optimization is able to implement.

Based on structural risk minimization principle, support vector machine (SVM) was first proposed by Cortes and Vapnik [10] in the 1990s. It has many advantages on classification, but multiple parameters have to be properly selected. Many research studies have been carried out on this topic. For a specific set of training samples, once classification accuracy is employed as objective function $\vec{x}$, solution vector $\vec{x}$ is formed by parameters of SVM, training of SVM classifier could be transformed into a multi-parameters optimization problem. Traditionally, most methods for SVM parameter optimization are based on grid search algorithm and genetic algorithm (GA) [11]. The recent focus is swarm intelligence algorithm-based methods, such as ant colony algorithm, particle swarm optimization (PSO) algorithm [12]. ABC algorithm is introduced for SVM parameter optimizing by Hsieh and Yeh [13]. Since multiple parameters of SVM-based soft-margin classifier need to be optimized, our D-ABC algorithm is highly suited for this purpose. Especially for multi-class classification problems, the length D of $\vec{x}$ is larger, and parameters including 'cost factor' of each class and kernel parameter are need to be optimized. Performance of classifier is evaluated by average classification accuracy after k-fold cross-validation. Experiments demonstrate that comparing with earlier ABC algorithms, our method have great improvement on convergence rate, and better parameters are obtained which lead to higher classification accuracy.

The main contributions of this article are (1) a modified ABC algorithm is proposed, named D-ABC algorithm; (2) D-ABC algorithm is applied to multi-parameters optimization of SVM soft-margin classifier. The article is organized as follows. In the following section, we introduce traditional ABC algorithm and several modified process along with their drawbacks. Moreover, description of D-ABC algorithm is presented. Multi-parameters optimization of SVM by D-ABC algorithm is illustrated in Section "Multi-parameters optimization of SVM-based soft-margin classifier", and accordingly experimental settings and analysis are stated. Finally, the last section concludes this study.

Methodology

Traditional ABC algorithm

ABC algorithm is inspired by the foraging behavior of real bee colony. The objective of a bee colony is to maximize the nectar amount stored in the hive. The mission is implemented by all the members of the colony, by efficient division of labor and role transforming. Each bee performs one of following three kinds of roles: employed bees (EB), onlooker bees (OB), and scout bees (SB). They could transform from one role to another in different phases of foraging. The flow of nectar collection is as follow:

1. In initial phase, there are only some SB and OB in the colony. SB are sent out to search for potential nectar source, and OB wait near the hive for being recruited. If any SB finds a nectar source, it will transform into EB.
2. EB collect some nectar and go back to the hive, and then dance with different forms to share information of the source with OB. Diverse forms of dance represent different quality of nectar source.
3. Each OB estimates quality of the nectar sources found by all EB, then follows one of EB to the corresponding source. All OB choose EB according to some probability. Better sources (more nectar) are more attractive (with larger probability to be selected) to OB.
4. Once any sources are exhausted, the corresponding EB will abandon them, transform into SB and search for new source.

In this way, the bee colony assigns more members to collect the better source and few members to collect the

ordinary ones. Thus, the nectar collection is more effective.

Analogously, in ABC algorithm, position of nectar source is presented by the coordinate in D-dimensional space. It is the solution vector $\vec{x}$ of some special problem, and the quality of nectar source is presented by the objective function $f(\vec{x})$ of this problem. Accordingly, optimization of this problem is implemented by simulating behaviors of the three kinds of bees. The flowchart of original ABC algorithm is shown in Figure 1. The main steps are as follow.

1. *Parameters initialization of ABC algorithm*. Population number (PN) and scout bee triggering threshold (*Limit*) are the key parameters of ABC algorithm. Maximum cycle number (MCN) or ideal fitness threshold (IFT) could be set for terminating algorithm. As stated in formula (1), all variables to be optimized form a D-dimensional vector $\vec{x}$. Restrict both upper bound (*UB*) and lower bound (*LB*) of each variable.
2. *Bee colony initialization*. In ABC algorithm, since SB transform into EB, they are not reckoned in PN. Generally, the initial nectar sources are found by PN/2 SB, and then they all transform into EB. The other PN/2 bees are OB. The initial PN/2 solutions are generated by formula (2) in principle. Specified initial value could be used only if needed. All further modifications are based on these PN/2 solutions, which is corresponding to the PN/2 EB.

$$x_i^{(j)} = LB^{(j)} + \phi_i^{(j)}\left(UB^{(j)} - LB^{(j)}\right) \quad \text{where } i = 1, 2, \ldots, PN/2 \quad j = 1, 2, \ldots, D \tag{2}$$

where $x_i^{(j)}$ is the jth elements of the ith solution. $\phi_i^{(j)}$ is uniformly distributed random real number in the range of [0, 1]. Objective function $f(\vec{x})$ is introduced to estimate the fitness of each solution $\vec{x}$. For parameter optimization of SVM classifier, $f(\vec{x})$ could be minimum classification error or maximum classification accuracy. Vector *Failure* is a counter, length PN/2, and is set to zero for counting optimizing failure of each EB.
3. Each cycle includes following phases:

    1) *Employed bee*: Each EB randomly modifies single element $x_i^{(j)}$ of source i by formula (3). Then fitness of the two solutions (before and after modification) is estimated. Greedy selection criterion is introduced to choose the one with better fitness, and the reserved one becomes new solution of this EB. If fitness of EB is not improved after modification, corresponding *Failure* counters will increase by 1.

$$\bar{x}_i^{(j)} = x_i^{(j)} + \lambda_i^{(j)}\left(x_i^{(j)} - x_k^{(j)}\right) \tag{3}$$

    where $x_i^{(j)}$ is defined as in formula (2), and $\bar{x}_i^{(j)}$ is the corresponding new element of the solution after modification. $\lambda_i^{(j)}$ is uniformly distributed random real number in the range of [−1, 1], and $x_k^{(j)}$ is the jth elements of $\vec{x}_k$. Note that $k \neq i$.

    2) *Estimate recruiting probability*. By formula (4), fitness and recruiting probability of each EB are calculated.

$$\begin{cases} \text{Fitness(i)} = \dfrac{1}{1 + f(\vec{x}_i)} \\ \text{prob(i)} = \dfrac{\text{Fitness(i)}}{\sum_{i=1}^{\text{PN}/2} \text{Fitness(i)}} \end{cases} \tag{4}$$

    3) *Onlooker bee*: 'roulette wheel' selection mechanism is introduced. It forces each OB following one of EB according recruiting probability. Owing to better solutions corresponding to larger recruiting probability, they obtain more chance to be optimized. Then each solution will be modified again by its followers (OB), using same steps as employed bee phase, from steps 1 to 6.
    4) *Record best solution*. All PN/2 solutions after modification are ranked according to their fitness, and best solution of current cycle is reserved. The termination conditions are then checked. When cycle counter reach the MCN or an ideal solution is found (reach IFT), the algorithm is over.
    5) *Scout bee*. If *Failure* counters of any solutions exceed *Limit*, the corresponding solution is abandoned, and scout bee is triggered. For example, if the lth solution is abandoned, a new solution is generated to replace the original one using formula (2), where set $i = l$.

By above operations, ABC algorithm performs optimization. Nevertheless, in both EB and OB phases, the algorithm merely modify single element of the solution in each cycle. If the length of the solution vector D is large, it makes inefficiency improvement in each cycle. In [2], MR is proposed, which is a real number factor in [0, 1]. For element $x_i^{(j)}$ of solution i, a uniformly distributed random real number ($0 \leq R_i^{(j)} \leq 1$) is produced. If $R_i^{(j)} \leq MR$, element $x_i^{(j)}$ will be modified and others not. Moreover, if all $R_i^{(j)}$

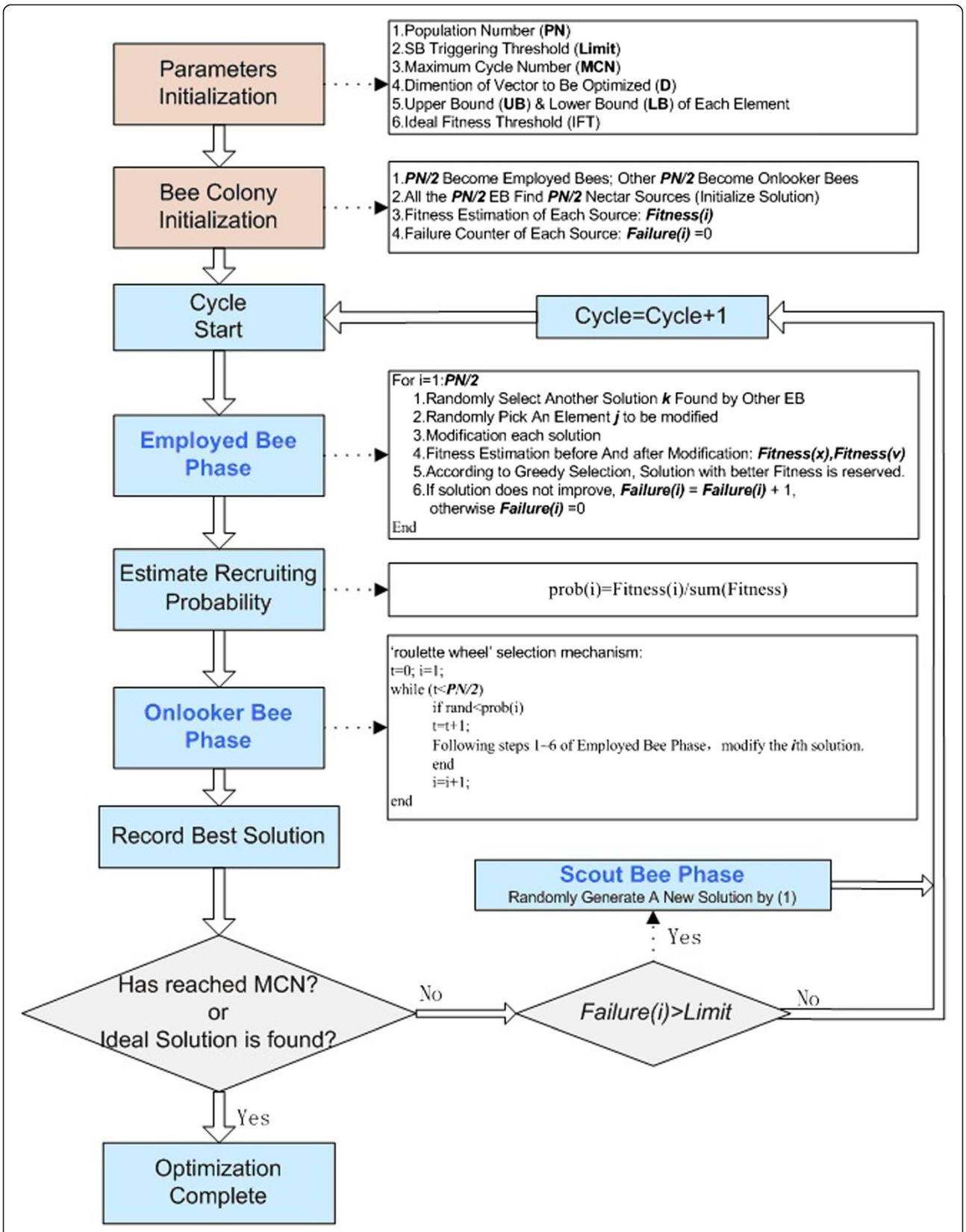


Figure 1 Flowchart of ABC algorithm. It presents flowchart of original ABC algorithm. Colorful frames show the main phases, and detailed introduction are shown in the followed frame.

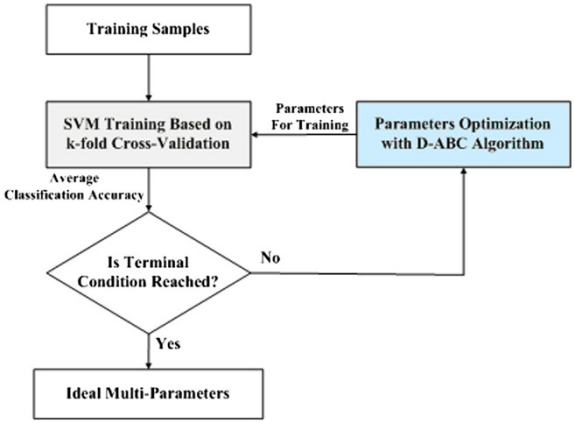


Figure 2 Flowchart of D-ABC algorithm-based multi-parameters optimization. It presents flowchart of optimizing process.

are larger than MR, ensure at least one parameter being modified by original algorithm. Although this MR-ABC algorithm improves the convergence rate of basic algorithm to some extent, its robustness is not ideal according to testing by abundant experiments.

D-ABC algorithm

The original idea of ABC algorithm is to perform hierarchical optimization. Overall, global searching is performed by EB and local searching is implemented by OB. However, this idea is not prominent in traditional ABC algorithms, because the modifying extent of EB and OB is similar and relatively fixed. Dynamic modifying extent is more reasonable. To achieve more effective optimization, the activity of bees must be dynamic in different stages of the algorithm. Our idea is that global searching should be dominant in early cycles and local searching should be primary in the posterior cycles. This could be more consistent with actions of real bees: EB become main force in the initial, then more and more OB follow, they play the major role afterwards. Specifically, in early stages of optimization, audaciously modify more elements of $\vec{x}$ in EB phase. That makes the bees approaching better solution by a greater probability. Furthermore, OB become active in posterior stages, and they modify more elements of $\vec{x}$. That provides more opportunities to jump out of local optimal solution.

Table 1 Range of parameters to be optimized

Items	Lower bound (*LB*)	Upper bound (*UB*)
'Cost factor' C	0.01	100
'Kernel Parameter' γ	0.001	100
'Weight Parameters for Each Class' q_j ($j = 1,2,\ldots,n$)	0	1

Table 2 Dataset declaration

Dataset	Number of samples	Number of features	Number of labels	*D*
1. Wine classification	178	13	3	5
2.Image segment	2310	19	7	9
3.Building classification (a)	600	8	30	32
4.Building classification (b)	1000	8	50	52

Consequently, we propose a dynamic 'activity' factor, and introduce it into modification operation of EB and OB phases. Adjust number of elements of the solution vector in each cycle. The 'activity' factor δ could be defined as following two forms by formulas (5) and (6), alternatively:

$$\delta = \Phi(C_c, MCN, D) \tag{5}$$

$$\delta = \Delta(F_c, IFT, D) \tag{6}$$

where C_c is current cycle number, , D is length of solution, F_c is current best fitness. The alternation of the two definitions depends on the termination condition of the algorithm. If using MCN to terminate the optimization, δ is defined as formula (5). And if IFT is employed, δ is defined as formula (6). δ_{EB} and δ_{OB} are 'activity' factor of EB and OB, respectively. Employ τ as the progress rate of the optimization, δ_{EB} and δ_{OB} subject to: (1) δ_{EB} grows with τ, when τ is not beyond half of total progress. $\delta_{EB} \in [0,1]$; (2) δ_{OB} grows with τ, when τ is beyond half of total progress. $\delta_{OB} \in [0,1]$. Explicit formulas could be determined according to specific problems. In this article, following scheme is suggested when utilize MCN as termination condition of the algorithm.

(1) In early stages, for EB phase, δ_{EB} is defined as formula (7). It reduces with C_c increasing, and N_{EB} elements are randomly picked to be modified; For OB phase, MR method is recommended. Audacious global modification and conservative local modification are implemented.

Table 3 Parameters settings of different optimization algorithm

Algorithm	*IFT*	*PN*	*MR*	*MCN*	*Limit*	Run times	*'k-fold' cross-validation*
PSO	100 %	10	—	100	—	20	10
ABC	100 %	10	—	100	100	20	10
MR-ABC	100 %	10	0.5	100	100	20	10
D-ABC	100 %	10	0.5	100	100	20	10

Table 4 Initial objection value of different datasets

Initial accuracy	EB1 (%)	EB2 (%)	EB3 (%)	EB4 (%)	EB5 (%)	EB6 (%)	EB7 (%)	EB8 (%)	EB9 (%)	EB10 (%)
Dataset 1	40.44	40.45	40.44	40.45	40.44	40.44	40.44	40.44	40.44	56.18
Dataset 2	30.95	24.76	21.90	23.81	28.57	24.28	30.95	21.90	21.90	22.86
Dataset 3	72.5	72.5	72	71.5	71	71.67	71.17	70.83	71.17	71.83
Dataset 4	65.1	60.7	65.7	66.4	66.7	65.7	66.1	66.5	64.3	67.6

$$\left.\begin{array}{l} EB \rightarrow N_{EB} = [\delta_{EB} \cdot D] = \left[\left(1 - \dfrac{C_c}{MCN}\right) \cdot D\right] \\ OB \rightarrow \mathrm{MR} \end{array}\right\} \quad if\ C_c \le \frac{MCN}{2} \tag{7}$$

(2) In posterior stages, for EB phase, MR method is reused; For OB phase, δ_{OB} is defined as formula (8). It increases with C_c growing, and N_{OB} elements are randomly picked to be modified. Conservative global modification and audacious local modification are implemented.

$$\left.\begin{array}{l} EB \rightarrow \mathrm{MR} \\ OB \rightarrow N_{OB} = [\delta_{OB} \cdot D] = \left[\dfrac{C_c}{MCN} \cdot D\right] \end{array}\right\} \quad if\ C_c > \frac{MCN}{2} \tag{8}$$

Furthermore, D-ABC algorithm is closely to the length D of solution vector. When D is small, there is practically little difference between original ABC algorithm and D-ABC algorithm. And for larger D, the advantages of D-ABC algorithm are prominent on convergence rate and improving the quality of solutions.

Multi-parameters optimization of SVM-based soft-margin classifier

Introduction of SVM parameters optimization

As we all know, training soft-margin classifier is a constrained optimization problem as formula (9). l is number of samples, x_i is ith sample, and y_i is the label of sample i.

$$\begin{aligned} &minimize && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\zeta_i \\ &subject\ to && y_i[wx_i + b] \ge 1 - \zeta_i\ (\zeta_i \ge 0, i = 1, 2, \ldots, l) \end{aligned} \tag{9}$$

It is a quadratic programming problem, which maximum the margin ($2/\|w\|$) when restricting the least classification error rate.

To solve unbalanced problem of training samples, 'slack variable' (ζ) and 'cost' factor (C) are introduced to process outlier samples and compromise the position of optimal separating hyper-plane. Large C indicates attaching importance to the loss of outliers of different classes. SVM needs to assign different C for each class. If these cost factors are not properly set, poor classification result will be obtained. However, experience-based setting is not robust. As a result, the

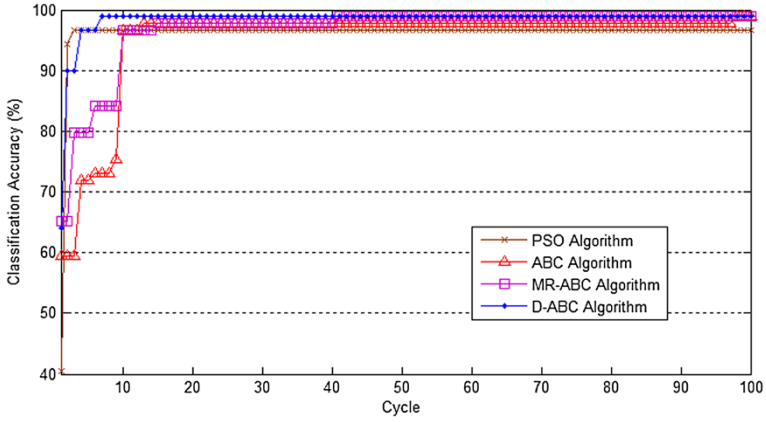


Figure 3 Performance comparison of different algorithms for Dataset 1. *X*-axis presents number of cycles, and *Y*-axis indicates classification accuracy after optimized by different algorithms.

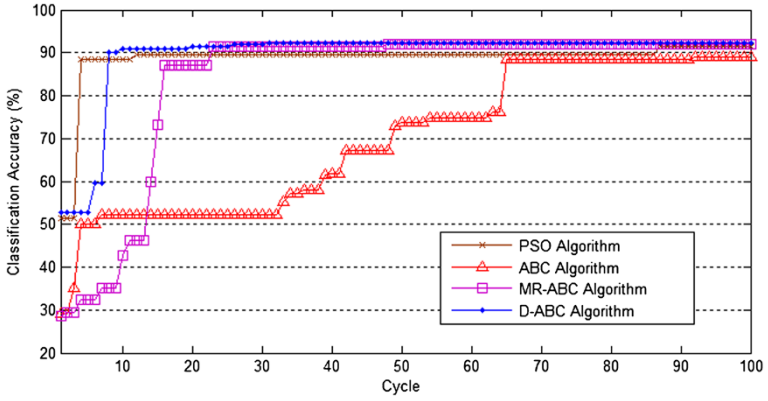


Figure 4 Performance comparison of different algorithms for Dataset 2. *X*-axis presents number of cycles, and *Y*-axis indicates classification accuracy after optimized by different algorithms.

multi-parameters optimizing problem needs to be solved, and parameters to be optimized will increase with number of class.

Additionally, parameters of kernel function of SVM need to be optimized. Solving (9) with Lagrange multiplier method, the separating classification function could be obtained as formulas (10) and (11), where α_i is Lagrange factor.

$$w = \sum_{i=1}^{l} (\alpha_i y_i x_i) \tag{10}$$

$$g(x) = \sum_{i=1}^{l} a_i y < x_i, x > + b \tag{11}$$

When samples are linearly inseparable, SVM processes nonlinear problem as linear classification in high-dimensional, which is performed by kernel function as formula (12). Both number and type of parameters to be optimized are determined by the kernel function.

$$g(x) = \sum_{i=1}^{l} a_i y K(x_i, x, \gamma) + b \tag{12}$$

All above parameters to be optimized compose a vector $\vec{x}$, and the multi-parameters optimization problem is

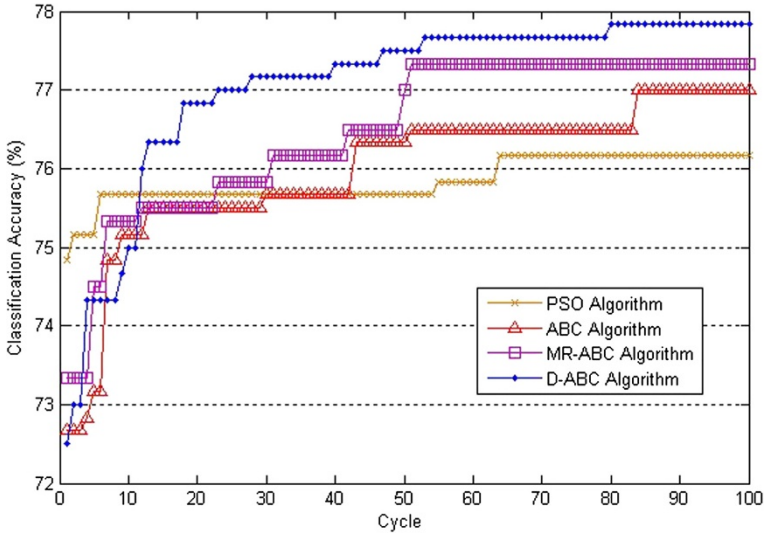


Figure 5 Performance comparison of different algorithms for Dataset 3. *X*-axis presents number of cycles, and *Y*-axis indicates classification accuracy after optimized by different algorithms.

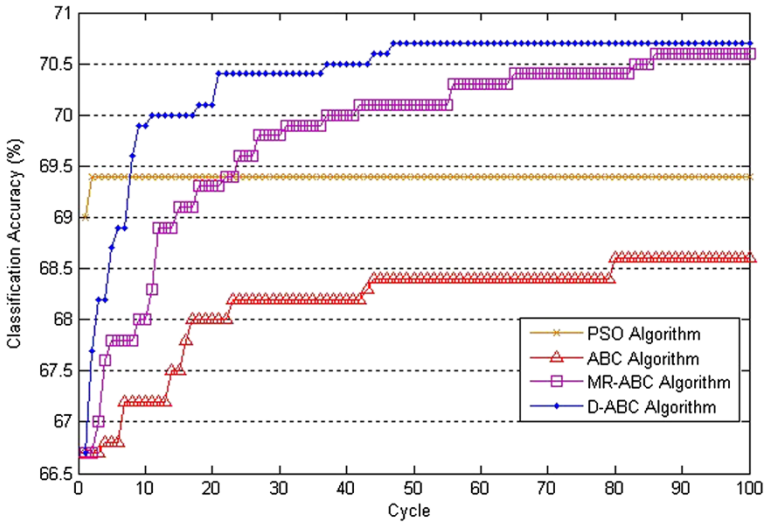


Figure 6 Performance comparison of different algorithms for Dataset 4. *X*-axis presents number of cycles, and *Y*-axis indicates classification accuracy after optimized by different algorithms.

defined as (13):

$$\begin{array}{ll} \text{maximize} & f(\vec{x}) = f\left(\{C, \gamma, q_1, q_2, \ldots, q_n\}\right) \\ \text{subject to} & y_i[wx_i + b] \geq 1 - \zeta_i \, (\zeta_i \geq 0, i = 1, 2, \ldots, l) \end{array} \tag{13}$$

where C is the cost factor, γ is the kernel parameter, and n is number of labels $q_j (j = 1, 2, \ldots, n)$, are weight parameters of each class, which set the cost factor C of class j to q, C. Moreover, to obtain creditable classification accuracy, 'k-fold' cross-validation is utilized for testing performance of SVM classifier. In our experiments, k is set to 10. Define the objective function $f(\vec{x})$ as the average classification accuracy of '10-fold' cross-validation as formula (14):

$$f(\vec{x}) = \sum_{m=1}^{k} Accuracy_m / k \tag{14}$$

Consequently, this problem could be solved by optimization algorithm. Owing to multiple parameters need to be optimized, our D-ABC algorithm is more suitable than traditional ABC algorithm. The flowchart of D-ABC

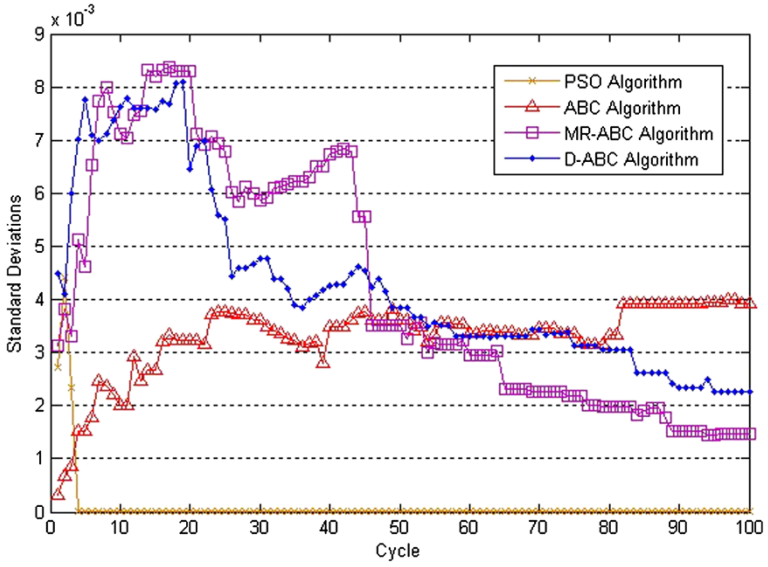


Figure 7 Standard deviations of different algorithms for Dataset 3. *X*-axis presents number of cycles, and *Y*-axis indicates standard deviations of classification accuracy after optimized by different algorithms for 20 times runs.

algorithm based multi-parameters optimization is shown in Figure 2.

For a set of training samples, D-ABC algorithm modifies parameters vector $\vec{x} = \{C, \gamma, q_1, q_2, \ldots, q_n\}$ cycle-by-cycle, and search best $\vec{x}$ for maximizing the classification accuracy.

Experiments

In this article, multi-class SVM-based soft-margin classifier is performed by C-support vector classification (C-SVC) toolbox. It is from LIBSVM toolbox supplied by Cheng and Lin [14]. The toolbox supply several typical kernel functions. Radial basis function is employed as kernel function in our experiments, and kernel parameter γ need to be optimized. For n-class classification, all parameters to be optimized and their range are presented in Table 1. Obviously, the length of vector $\vec{x}$ is $D = n + 2$. The dataset utilized for SVM training is as Table 2 shows. 'Wine' and 'Image Segment' are two typical testing dataset, which are widely used for testing SVM-based classifier. The two 'building' datasets are collected by us especially for multi-parameters optimization problem.

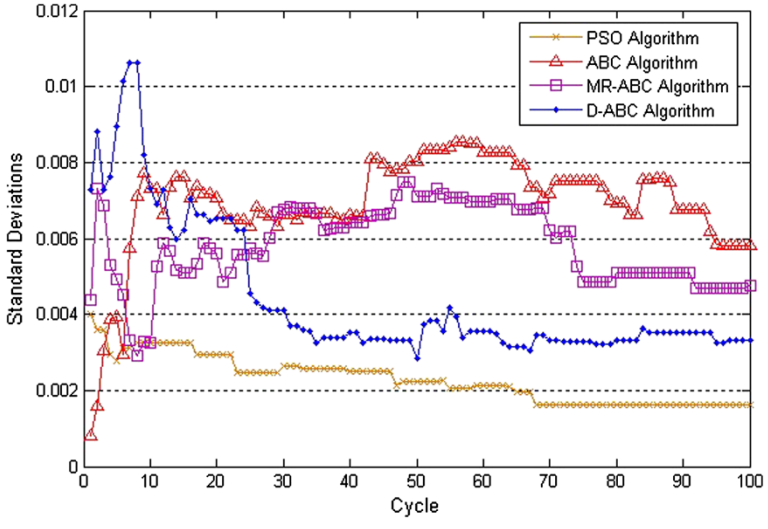
Performances of PSO algorithm, original ABC algorithm, MR-ABC algorithm, and D-ABC algorithm are compared for this optimization problem. All algorithms are coded under MATLAB 2011b. Main hardware configuration of our computer: Intel®Core(TM)2 Duo CPU P8400@2.26 GHz 2.27 GHz, 2.00 GB RAM.

According to principle of fair comparison: (1) corresponding initialization parameters are same set in these algorithms as Table 3 shows,the settings are according to [2]; (2) using same starting searching points to initialize the colony, as shown in Table 4 and Appendix. Mean value of 20 times running by different algorithms are collected as the final results for the four datasets, which are shown in Figures 3, 4, 5, and 6 and Table 4. Particularly, to verify the robustness of different algorithm, standard deviations of the 20 times run are given by Figures 7 and 8, for the two high-dimensional datasets.

Table 5 Best solution obtained by different optimization algorithms

Algorithm	Dataset 1 (%)	Dataset 2 (%)	Dataset 3 (%)	Dataset 4 (%)
PSO	96.63	91.43	76.33	69.40
ABC	98.88	89.05	77.00	68.60
MR-ABC	98.88	91.90	77.33	70.60
D-ABC	98.88	92.38	77.83	70.70

Note that we choose measuring the convergence rate in cycle for following reasons: Generally, computational time of calling objection function is much larger than other parts of ABC algorithm, particularly when our objection function includes multiple times SVM training. The SVM training takes more than much time, and in each cycle, the code of objection function will be called many times (same times in each cycle for different algorithm, and times is determined by parameter PN). Objection function calling occupies more than 90% computational time of both original ABC algorithm and modified ones (for instance, for dataset 4, about 22.3 s



"

Figure 8 Standard deviations of different algorithms for Dataset 4. *X*-axis presents number of cycles, and *Y*-axis indicates standard deviations of classification accuracy after optimized by different algorithms for 20 times runs.

are cost by running D-ABC algorithm for each cycle, 21.7 s for MR-ABC algorithm, and 20.5 s for original ABC algorithm. It is obviously that objection function cost most time in each cycle). Moreover, every time the objection function is called, the solution is modified. Therefore, it is more reasonable measuring the convergence rate in cycle than in time. On the contrary, if measuring the convergence rate in specified computational time, each solution might be modified different times, which could be unfair.

As is shown in Figure 3, for dataset 1, D-ABC algorithm rapidly find a solution $\vec{x}$, with which SVM could best training the data and obtain a classification accuracy of 98.88 %, while original ABC algorithm and MR-ABC obtain that solution slowly. Though PSO algorithm has a good convergence rate, it could not get an ideal solution. As is shown in Figure 4, for dataset 2, similarly, compared with original ABC algorithm and MR-ABC, D-ABC algorithm performs better convergence rate and obtains higher classification accuracy, and the improvement is more obvious than dataset 1. PSO algorithm still converges fast but an unsatisfactory solution. By contrast, D-ABC algorithm obtains a classification accuracy of 92.38 %.

The results above have demonstrated the advantages of D-ABC algorithm for lower dimension of parameter-vector like datasets 1 and 2. Furthermore, datasets 3 and 4 are collected for testing optimization of higher dimensional $\vec{x}$. As is shown in Figures 5, 6, and Table 5, similar conclusions could be obtained that D-ABC algorithm has certain advantages over other algorithms, which lead to greater improvement on convergence rate and quality of solution, especially when D is larger. Moreover, standard deviations (of 20 times run) for the two groups of datasets are shown in Figures 7 and 8, respectively. The curves illustrate the standard deviations of objection function in different optimizing cycles, and the relatively lower standard deviations have been obtained by D-ABC algorithm for both datasets 3 and 4, which indicates that D-ABC algorithm has good robustness.

Conclusion and discussion

In this article, two parts of work have been studied. First, D-ABC algorithm is introduced to improve the disadvantages of traditional ABC algorithms: poor convergence rate and local optimizing. Second, D-ABC algorithm is utilized for multi-parameters optimization of SVM classifier. Experiments results demonstrate that D-ABC algorithm is in many ways superior to traditional ABC algorithms. It effectively ameliorates the convergence rate and local optimum. Typically, for multi-parameters optimization, when length D of vector (to be optimized) is larger, our study has provided substantial evidence for the advantages of D-ABC algorithm on quality of solution and convergence rate. When D-ABC algorithm is employed for optimizing multi-parameters of SVM-based soft-margin classifier, great improvement is obtained on performance of the classifier. Moreover, the robustness of D-ABC algorithm is proofed. Furthermore, the idea of D-ABC algorithm could be associated with other modified ABC algorithms, whose modification is on other phase of original ABC algorithm, and it might further improve traditional ABC algorithm in future work.

Appendix

The starting searching points of the four group of experiments are tabulated in Tables 6, 7, 8, and 9.

Table 6 The initial colony of dataset 1

EB1	EB2	EB3	EB4	EB5	EB6	EB7	EB8	EB9	EB10
27.21815	86.80153	74.1761	44.84249	70.99292	94.4387	17.49438	24.53517	64.12882	80.88039
853.3725	398.1237	115.5027	80.29015	360.474	828.9073	214.6176	791.0424	654.6918	26.15619
0.785776	0.922563	0.492313	0.834012	0.131354	0.759783	0.925736	0.832708	0.259401	0.213022
0.522315	0.397357	0.47911	0.993904	0.604478	0.944909	0.490442	0.437947	0.772656	0.744067
0.442904	0.053	0.087822	0.797986	0.655582	0.032336	0.557067	0.719802	0.110408	0.216647

Table 7 The initial colony of dataset 2

EB1	EB2	EB3	EB4	EB5	EB6	EB7	EB8	EB9	EB10
82.97033	84.92364	37.31617	59.35914	87.268	93.35681	66.87958	20.75697	65.41967	7.29795
406.7328	666.9349	933.7263	810.9519	484.5534	756.7516	417.0533	971.7863	987.9748	864.1489
0.388884	0.454742	0.246687	0.784423	0.882838	0.913712	0.558285	0.598868	0.148877	0.899713
0.450394	0.205672	0.899651	0.762586	0.882486	0.28495	0.673226	0.66428	0.122815	0.407318
0.275287	0.71667	0.283384	0.896199	0.826579	0.390027	0.497903	0.694805	0.834369	0.60963
0.574737	0.326042	0.456425	0.713796	0.884405	0.720856	0.018613	0.674776	0.438509	0.43782
0.117037	0.814682	0.324855	0.246228	0.342713	0.375692	0.546554	0.56192	0.395822	0.398131
0.515367	0.657531	0.950915	0.722349	0.40008	0.831871	0.134338	0.060467	0.084247	0.163898
0.32422	0.301727	0.011681	0.539905	0.095373	0.146515	0.631141	0.85932	0.974222	0.570838

Table 8 The initial colony of dataset 3

EB1	EB2	EB3	EB4	EB5	EB6	EB7	EB8	EB9	EB10
81.4909	90.58861	12.78598	91.34625	63.27269	9.844286	27.92197	54.73346	95.75493	96.49236
157.6215	970.5931	957.1674	485.3808	800.2825	141.8949	421.7671	915.7364	792.2094	959.4928
0.655741	0.035712	0.849129	0.933993	0.678735	0.75774	0.743132	0.392227	0.655478	0.171187
0.706046	0.031833	0.276923	0.046171	0.097132	0.823458	0.694829	0.317099	0.950222	0.034446
0.438744	0.381558	0.765517	0.7952	0.186873	0.489764	0.445586	0.646313	0.709365	0.754687
0.276025	0.679703	0.655098	0.162612	0.118998	0.498364	0.959744	0.340386	0.585268	0.223812
0.751267	0.255095	0.505957	0.699077	0.890903	0.959291	0.547216	0.138624	0.149294	0.257508
0.840717	0.254282	0.814285	0.243525	0.929264	0.349984	0.196595	0.251084	0.616045	0.473289
0.35166	0.830829	0.585264	0.549724	0.917194	0.285839	0.7572	0.753729	0.380446	0.567822
0.075854	0.05395	0.530798	0.779167	0.934011	0.129906	0.568824	0.469391	0.011902	0.337123
0.162182	0.794285	0.311215	0.528533	0.165649	0.601982	0.262971	0.654079	0.689215	0.748152
0.450542	0.083821	0.228977	0.913337	0.152378	0.825817	0.538342	0.996135	0.078176	0.442678
0.106653	0.961898	0.004634	0.77491	0.817303	0.868695	0.084436	0.399783	0.25987	0.800068
0.431414	0.910648	0.181847	0.263803	0.145539	0.136069	0.869292	0.579705	0.54986	0.144955
0.853031	0.622055	0.350952	0.51325	0.401808	0.075967	0.239916	0.123319	0.183908	0.239953
0.417267	0.049654	0.902716	0.944787	0.490864	0.489253	0.337719	0.900054	0.369247	0.111203
0.780252	0.389739	0.241691	0.403912	0.096455	0.131973	0.942051	0.956135	0.575209	0.05978
0.23478	0.353159	0.821194	0.015403	0.043024	0.16899	0.649115	0.731722	0.647746	0.450924
0.547009	0.296321	0.744693	0.188955	0.686775	0.183511	0.368485	0.625619	0.780227	0.081126
0.929386	0.775713	0.486792	0.435859	0.446784	0.306349	0.508509	0.510772	0.817628	0.794831
0.644318	0.378609	0.81158	0.532826	0.350727	0.939002	0.875943	0.550156	0.622475	0.587045
0.207742	0.301246	0.470923	0.230488	0.844309	0.194764	0.225922	0.170708	0.227664	0.435699
0.311102	0.92338	0.430207	0.184816	0.904881	0.979748	0.43887	0.111119	0.258065	0.40872
0.594896	0.262212	0.602843	0.711216	0.221747	0.117418	0.296676	0.318778	0.424167	0.507858
0.085516	0.262482	0.801015	0.02922	0.928854	0.730331	0.488609	0.578525	0.237284	0.458849
0.963089	0.546806	0.521136	0.231594	0.488898	0.62406	0.679136	0.395515	0.367437	0.987982
0.037739	0.885168	0.913287	0.796184	0.098712	0.261871	0.335357	0.679728	0.136553	0.721227
0.106762	0.653757	0.494174	0.779052	0.715037	0.903721	0.890923	0.334163	0.698746	0.19781
0.030541	0.744074	0.500022	0.479922	0.904722	0.609867	0.617666	0.859442	0.805489	0.576722
0.182922	0.239932	0.886512	0.028674	0.489901	0.167927	0.978681	0.712694	0.500472	0.471088
0.059619	0.681972	0.042431	0.071445	0.52165	0.09673	0.818149	0.817547	0.72244	0.149865
0.659605	0.518595	0.972975	0.648991	0.800331	0.453798	0.432392	0.825314	0.08347	0.133171

Table 9 The initial colony of dataset 4

EB1	EB2	EB3	EB4	EB5	EB6	EB7	EB8	EB9	EB10
17.42152	39.15469	83.15484	80.3561	6.141071	39.98585	52.7349	41.73827	65.7203	62.83454
291.9912	431.6569	15.49697	984.0639	167.1767	106.2253	372.416	198.1264	489.6927	339.5
0.95163	0.920332	0.052677	0.737858	0.269119	0.422836	0.547871	0.942737	0.417744	0.983052
0.301455	0.701099	0.666339	0.539126	0.698106	0.666528	0.178132	0.128014	0.99908	0.171121
0.032601	0.5612	0.881867	0.669175	0.190433	0.368917	0.460726	0.981638	0.156405	0.855523
0.644765	0.376272	0.190924	0.428253	0.482022	0.120612	0.589507	0.226188	0.384619	0.582986
0.251806	0.290441	0.617091	0.265281	0.824376	0.982663	0.730249	0.343877	0.584069	0.107769
0.906308	0.879654	0.817761	0.260728	0.594356	0.022513	0.425259	0.312719	0.161485	0.178766
0.422886	0.094229	0.598524	0.470924	0.695949	0.699888	0.638531	0.033604	0.068806	0.3196

Table 9 The initial colony of dataset 4 *(Continued)*

0.530864	0.654446	0.407619	0.819981	0.718359	0.968649	0.531334	0.325146	0.105629	0.610959
0.778802	0.423453	0.090823	0.266471	0.153657	0.281005	0.440085	0.527143	0.457424	0.875372
0.518052	0.943623	0.637709	0.957694	0.240707	0.676122	0.289065	0.671808	0.69514	0.067993
0.25479	0.22404	0.667833	0.844392	0.344462	0.78052	0.675332	0.006715	0.60217	0.386771
0.915991	0.001151	0.462449	0.424349	0.460916	0.77016	0.322472	0.784739	0.471357	0.035763
0.175874	0.721758	0.473486	0.152721	0.341125	0.607389	0.191745	0.738427	0.24285	0.917424
0.269062	0.7655	0.188662	0.287498	0.091113	0.576209	0.683363	0.546593	0.425729	0.644443
0.647618	0.679017	0.635787	0.945174	0.208935	0.709282	0.236231	0.119396	0.607304	0.450138
0.458725	0.661945	0.770286	0.350218	0.66201	0.416159	0.841929	0.832917	0.256441	0.613461
0.582249	0.540739	0.869941	0.264779	0.318074	0.119215	0.939829	0.645552	0.479463	0.639317
0.544716	0.647311	0.543886	0.721047	0.522495	0.993705	0.218677	0.105798	0.109697	0.063591
0.40458	0.448373	0.365816	0.763505	0.627896	0.77198	0.932854	0.972741	0.192028	0.138874
0.696266	0.09382	0.525404	0.530344	0.86114	0.484853	0.393456	0.671431	0.741258	0.520052
0.347713	0.149997	0.586092	0.262145	0.044454	0.754933	0.242785	0.442402	0.687796	0.359228
0.73634	0.394707	0.683416	0.704047	0.442305	0.019578	0.330858	0.424309	0.27027	0.197054
0.821721	0.429921	0.887771	0.391183	0.769114	0.396792	0.808514	0.755077	0.377396	0.216019
0.790407	0.949304	0.327565	0.671264	0.438645	0.833501	0.768854	0.167254	0.86198	0.989872
0.514423	0.884281	0.588026	0.154752	0.199863	0.406955	0.748706	0.825584	0.789963	0.318524
0.534064	0.089951	0.111706	0.136293	0.678652	0.495177	0.18971	0.495006	0.147608	0.054974
0.850713	0.56056	0.929609	0.696667	0.582791	0.815397	0.879014	0.988912	0.000522	0.865439
0.612566	0.98995	0.52768	0.479523	0.801348	0.227843	0.498094	0.900852	0.574661	0.845178
0.73864	0.585987	0.246735	0.666416	0.083483	0.62596	0.660945	0.729752	0.890752	0.982303
0.769029	0.581446	0.928313	0.58009	0.016983	0.12086	0.862711	0.484297	0.844856	0.209405
0.552291	0.629883	0.031991	0.614713	0.362411	0.049533	0.48957	0.19251	0.123084	0.205494
0.146515	0.189072	0.042652	0.635198	0.281867	0.538597	0.695163	0.499116	0.535801	0.445183
0.123932	0.490357	0.852998	0.873927	0.270294	0.208461	0.56498	0.640312	0.417029	0.205976
0.947933	0.082071	0.105709	0.142041	0.16646	0.620959	0.57371	0.052078	0.931201	0.728662
0.737842	0.063405	0.860441	0.934405	0.984398	0.858939	0.785559	0.513377	0.177602	0.398589
0.133931	0.03089	0.939142	0.301306	0.295534	0.332936	0.467068	0.648198	0.025228	0.842207
0.559033	0.8541	0.347879	0.446027	0.054239	0.177108	0.662808	0.330829	0.898486	0.118155
0.988418	0.539982	0.706917	0.999492	0.287849	0.414523	0.46484	0.763957	0.818204	0.100222
0.178117	0.359635	0.056705	0.521886	0.335849	0.175669	0.208947	0.905154	0.675391	0.468468
0.912132	0.104012	0.745546	0.736267	0.561861	0.184194	0.597211	0.299937	0.134123	0.212602
0.894942	0.071453	0.242487	0.053754	0.441722	0.013283	0.897191	0.196658	0.093371	0.307367
0.456058	0.101669	0.99539	0.332093	0.297347	0.062045	0.298244	0.046351	0.505428	0.761426
0.63107	0.089892	0.080862	0.777241	0.905135	0.533772	0.109154	0.825809	0.338098	0.293973
0.746313	0.010337	0.048447	0.667916	0.603468	0.526102	0.729709	0.707253	0.781377	0.287977
0.692532	0.55667	0.396521	0.061591	0.780176	0.337584	0.607866	0.741254	0.104813	0.127888
0.54954	0.485229	0.890476	0.79896	0.734341	0.051332	0.072885	0.088527	0.798351	0.943008
0.683716	0.132083	0.722725	0.110353	0.117493	0.640718	0.328814	0.653812	0.749131	0.583186
0.740032	0.234827	0.734958	0.970599	0.86693	0.086235	0.366437	0.369199	0.685028	0.597942
0.789364	0.367653	0.206028	0.086667	0.771934	0.205675	0.388272	0.551779	0.228953	0.641941
0.48448	0.151846	0.781932	0.100606	0.294066	0.237373	0.530872	0.091499	0.405315	0.104846

Abbreviations
ABC: Artificial bee colony; D-ABC: Dynamic artificial bee colony; EB: Employed bees; PSO: Particle swarm optimization; RSEM: Randomly single element modification; IFT: Ideal fitness threshold; LB: Lower bound; MCN: Maximum cycle number; MR: Modification rate; MR-ABC: Modification rate artificial bee colony; OB: Onlooker bees; PN: Population number; SB: Scout bees; SVM: Support vector machine; UB: Upper bound.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of information engineering, Harbin institute of technology, (92 West Dazhi Street), Harbin (150001), China. [2]Department of information engineering, Harbin institute of technology, (92 West Dazhi Street), Harbin (150001), China. [3]Department of modern control, Institute of automation of Heilongjiang academy of Science, (Hanshui road), Harbin (150090), China.

Received: 6 March 2012 Accepted: 6 July 2012
Published: 24 July 2012

References

1. D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report, TR06* (Erciyes University Press, Erciyes, 2005)
2. D. Karaboga, B. Akay, A modified artificial bee colony (ABC) algorithm for constrained optimization problems. Appl. Soft Comput. **11**(3), 3021–3031 (2010). doi:10.1016/j.asoc.2010.12.001
3. B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization. Inf. Sci **192**, 120–142 (2012). doi:10.1016/j.ins.2010.07.015
4. A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Appl. Soft Comput. **9**(2), 625–631 (2010). doi:10.1016/j.asoc.2008.09.001
5. L. Bao, J. Zeng, *Comparison and analysis of the selection mechanism in the artificial bee colony algorithm*, 1st edn. (Paper presented at Ninth International Conference on Hybrid Intelligent Systems (HIS '09), Shenyang, LiaoNing, China, 2009), pp. 411–416
6. R.S. Parpinelli, C.M.V. Benitez, H.S. Lopes, Parallel approaches for the artificial bee colony algorithm, handbook of swarm intelligence: concepts. Princ. Appl. **8**, 329 (2011)
7. D. Haijun, F. Qingxian, Artificial bee colony algorithm based on Boltzmann selection policy. Comput. Eng. Appl. **45**(31), 53–55 (2009)
8. P. Tsai, J. Pan, B. Liao, S. Chu, *Interactive artificial bee colony (IABC) optimization* (ISI2008, Taipei Taiwan, 2008)
9. R. Akbari, A. Mohammadi, K. Ziarati, A novel bee swarm optimization algorithm for numerical function optimization. Commun. Nonlinear Sci. Numer. Simul **15**, 3142–3155 (2010). doi:10.1016/j.cnsns.2009.11.003
10. C. Cortes, V. Vapnik, Support-vector networks. Mach. Learn **20**, 273–297 (1995). doi:10.1007/BF00994018
11. F. Samadzadegan, A. Soleymani, R.A. Abbaspour, *Evaluation of Genetic Algorithms for tuning SVM parameters in multi-class problems* (Paper presented at 11th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, 2010), pp. 323–328
12. L. Yang, H.T. Wang, *Classification based on particle swarm optimization for least square support vector machines training* (Paper presented at Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI), Jinggangshan, 2010), pp. 246–249
13. T.J. Hsieh, W.C. Yeh, Knowledge discovery employing grid scheme least squares support vector machines based on orthogonal design bee colony algorithm. IEEE Trans. Syst. Man Cybern. B: Cybernetics **41**(5), 1–15 (2011). doi:10.1109/TSMCB.2011.2116007
14. C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 27 (2011). doi:10.1145/1961189.1961199

doi:10.1186/1687-6180-2012-160
Cite this article as: Yan *et al.*: Dynamic artificial bee colony algorithm for multi-parameters optimization of support vector machine-based soft-margin classifier. *EURASIP Journal on Advances in Signal Processing* 2012 **2012**:146.