Research Article

Mi Li, Ming Zhang, Huan Chen, Shengfu Lu*

A Method of Biomedical Information Classification based on Particle Swarm Optimization with Inertia Weight and Mutation

https://doi.org/10.1515/biol-2018-0044
Received March 23, 2018; accepted June 7, 2018

Abstract: With the rapid development of information technology and biomedical engineering, people can get more and more information. At the same time, they begin to study how to apply the advanced technology in biomedical information. The main research of this paper is to optimize the machine learning method by particle swarm optimization (PSO) and apply it in the classification of biomedical data. In order to improve the performance of the classification model, we compared the different inertia weight strategies and mutation strategies and their combinations with PSO, and obtained the best inertia weight strategy without mutation, the best mutation strategy without inertia weight and the best combination of the two. Then, we used the three PSO algorithms to optimize the parameters of support vector machine in the classification of biomedical data. We found that the PSO algorithm with the combination of inertia weight and mutation strategy and the inertia weight strategy that we proposed could improve the classification accuracy. This study has an important reference value for the prediction of clinical diseases.

Keywords: Biomedical information classification; Support vector machine; Particle swarm optimization; Inertia weight strategy; Mutation strategy

***Corresponding author: Shengfu Lu,** Beijing Advanced Innovation Center for Future Internet Technology, Beijing University of Technology, Beijing 100124, China, E-mail: lusf@bjut.edu.cn
Mi Li, Ming Zhang, Huan Chen, Shengfu Lu, Department of Automation, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
Mi Li, Ming Zhang, Huan Chen, Shengfu Lu, The Beijing International Collaboration Base on Brain Informatics and Wisdom Services, Beijing 100024, China

1 Introduction

Bioinformatics integrates computer science technology and biological information technology, which reveals the significance of biological research and applications. An important part of bioinformatics is to predict which category it belongs by the given data. The rapid development of computer technology has brought much biological information. It is very difficult for people to intuitively visualize these large amounts of data. In recent years, many scholars have applied machine learning algorithms to predict diseases in the field of biomedicine and achieved good results. The common machine learning methods include support vector machine (SVM) proposed by Joachims [1], the decision tree proposed by Quinlan [2], the *k* nearest proposed by Fukunaga and Narendra [3], the Bayesian algorithm proposed by Bataineh and Al-Qudah [4] and the deep learning proposed by Lecun and Bengio [5], etc. Since SVM has a great advantage in solving small sample, nonlinear and high-dimensional problems, SVM has been widely used as a core method in biomedicine classification and recognition. For example, Chen and Huang used the EEG signal to identify epilepsy with SVM [6]. Soares and Paiva used SVM to diagnose breast tumor mass [7]. Zhou and Cui used SVM and Bayesian algorithms to predict protein localization [8]. Qu and Chen used SVM for the segmentation of a pathological picture of breast cancer tissue [9]. Mishra and Lakkadwala used SVM to predict cardiovascular disease [10]. Liu and Zhang used SVM to predict osteosarcoma [11]. Parikh and Shah used SVM to diagnose skin disease [12].

SVM is a supervised learning algorithm, and its basic idea is transforming the input space into a higher dimensional feature space through nonlinear transformation, and then the optimal classification hyperplane is calculated in this space, so as to classify the unknown sample. SVM is widely used in the field of machine learning. In order to further improve

the performance of SVM, many variants of support vector machine are proposed. For example, Fung and Mangasarian proposed a proximal support vector machine (PSVM) [13]. The advantage of PSVM relative to SVM is that it uses an equation set to replace the convex programming problem in SVM. Under the condition of maximum interval, the PSVM uses two parallel hyperplanes to fit the two types of samples. Lin and Wang proposed a fuzzy support vector machine (FSVM) [14]. According to the different contribution of input samples, FSVM gives the samples different membership degrees, and separated the noise from effective samples. Although FSVM improved the traditional SVM, it is difficult to determine the membership function in FSVM at present. Jayadeva and Khemchandani proposed the twin support vector machine (TWSVM) [15]. TWSVM transforms two generalized feature problems into two smaller convex programming problems similar to support vector machine. SVM puts all the samples in the constraint condition of the convex programming problem, while TWSVM puts the samples that are opposite to the target function in the constraint conditions, so that the training speed of TWSVM is greatly improved. However, TWSVM also has the characteristics of lack of sparsity and low generalization ability, and it also needs improvement.

The most important part of support vector machine is the kernel function and its parameter, which can affect the performance of the SVM directly, thus the choice of kernel function is a key problem in SVM. The common kernel functions are linear kernel function, polynomial kernel function and radial basis kernel function (RBF). The linear kernel function is mainly used in linear separable situations. The RBF is mainly used in linear inseparable situations. Relative to the linear kernel function, the RBF kernel function can map the feature to a higher dimensional space, and the linear kernel function can be regarded as a special case of RBF. In general, the RBF kernel is widely used. For example, Kuo and Ho used the SVM with the RBF kernel in image classification [16]. Prabin and Veerappan used the SVM with mixed RBF kernel to diagnose the MRI image of brain tumors [17]. Bousseta and Tayeb used SVM with RBF kernel in EEG data classification [18].

When using the SVM based on RBF kernel function to classify, we need to set the penalty factor (C) and the kernel parameter (g). Keerthi and Lin researched the penalty factor and the RBF kernel parameter in SVM [19], and analyzed the influence of different kernel parameters on the performance of the classifier. Chapelle and Vapnik proposed a method to adjust the kernel parameters of SVM automatically [20], but this method needs to compute the gradient of various bounds, which increases the complexity of the algorithm. In recent years, many swarm intelligent evolutionary algorithms have been proposed. Kennedy and Eberhart proposed the particle swarm optimization (PSO) algorithm [21], which simulates the process of bird hunting. Goldberg proposed the genetic algorithm (GA) based on the genetic process of nature [22]. Qinghong and Zhang proposed the ant colony algorithm according to the process of ants foraging [23]. Selima and Alsultan proposed the simulated annealing algorithm based on the process of temperature drop during solid annealing [24]. Eskandar and Sadollah proposed a water cycle algorithm (WCA) based on the process of water circulation in nature and the flow of rivers to the sea [25]. This algorithm shows good performance in solving the constraints problem, but the efficiency of the algorithm is not so high. In order to further improve the performance of the WCA algorithm, Pahnekolaie and Alfi proposed a gradient based water cycle algorithm, and applied the algorithm in the chaos suppression problem [26]. Gonçalves and Lopez proposed a search group algorithm (SGA) and applied it to truss structure optimization [27]. The SGA algorithm has a good ability of exploration and exploitation, but it is also sensitive to parameters. Seyedeh and Alireza proposed a fuzzy logic method to control the parameters of the SGA algorithm adaptively [28]. Zong and Kim proposed harmony search algorithm (HS) [29], which simulates musicians relying on their own memory to adjust the tone of each musical instrument in the band to achieve a wonderful state of harmony. Ameli and Alfi proposed a discrete harmony search algorithm (DHS), and applied it in the optimization of the capacitor position in the distribution system, and achieved a good result [30]. In these heuristic search algorithms, the PSO algorithm is relatively simple and easy to implement, and therefore is widely used in various optimization problems. For example, Zhang and Lv proposed an adaptive inertia weight chaotic PSO algorithm and applied it to train the single hidden layer neural network, and achieved a good classification result [31]. Wang and Phillips proposed the binary particle swarm optimization with mutation (BPSO-M), the binary particle swarm optimization with time-varying learning factors (BPSO-T) and the binary particle swarm optimization with the combination of mutation and time-varying learning factors (BPSO-MT), and applied these three algorithms to select features [32]. Zhang and Wang analyzed the status of swarm intelligent algorithms in detail, and summarized their applications in various industries [33]. Fernandez and Caballero used genetic algorithms to optimize the parameters of SVM [34]. Behravan and Dehghantanha used the binary particle

swarm optimization (BPSO) to select features [35]. Kuang and Zhangused the improved chaotic PSO algorithm to optimize the parameters of the mixed kernel function [36]. Subasi thought that PSO had a significant improvement in SVM [37]. Ren and Hu used a grid algorithm to optimize kernel function parameters [38]. Zhang thought that PSO was more efficient in coding and optimizing than the genetic algorithm [39]. Therefore, it is relatively common to use the PSO algorithm to optimize the parameters of SVM.

Because the original PSO algorithm easily falls into local optimal solutions, many scholars have improved the PSO algorithm. Tanweer and Auditya proposed a self-adjusting particle swarm algorithm [40]. Meng and Li adopted a cross search strategy and proposed a new updating formula, and horizontal crossover and vertical crossover operator are used [41]. Wang and Liu introduced a chaos search method and promoted the algorithm to jump out of the local optimal solution through the uncertainty of chaos [42]. Chen and Zhang adopted dynamic topology PSO algorithm in which the structure of the population is changed with the information of particles [43]. Liang and Kang proposed an adaptive mutation strategy and adopted a nonlinear variable inertia weight strategy [44]. The inertia weight of the PSO algorithm is an important parameter, and Shi and Eberhart introduced the inertia weight before the velocity term in the basic PSO algorithm [45]. Alireza and Modares proposed an adaptive inertia weight strategy. The improved PSO algorithm was applied to the optimization of PID parameters, and a good result was obtained compared with the genetic algorithm [46]. Eberhart and Shi proposed a random changing inertia weight strategy [47]. Malik and Rahman uses a sigmoid changing inertial weight strategy [48]. The inertia weight is stable at the beginning and end stages, and changes faster in the middle stage. Gholamian proposed a chaotic changing inertia weight [49]. Javad and Mousa proposed a nonlinear changing inertia weight [50]. Alireza and Fateh proposed a mutation strategy to improve the ability of global exploration and the speed of convergence, and used it to identify the hydraulic suspension system [51]. At present, the improvement methods of PSO algorithm mainly include two aspects: the inertia weight strategy and the mutation strategy. But most of the current strategies are unilateral, and there is almost no literature to compare the combinatorial performance of the mutation strategy and inertia weight strategy. In this paper, we compared the combinatorial performance of different inertia weights and mutation strategies, and got the best mutation strategy, the best inertia weight strategy and the best combination of them. Finally, we used the best mutation strategy, the best inertia weight strategy and the best combinations of PSO algorithms to optimize the parameters of SVM in biomedical information classification.

2 Algorithm and Model

2.1 Support Vector Machine

SVM is a machine learning method proposed in the 1990s based on statistical learning theory [52]. It is based on the principle of structural risk minimization. The kernel function method of SVM is used to transform the inseparable problem in low dimensional space into linearly separable problem in high dimensional space.

For a given sample set of SVM (xi, yi), i=1, 2,..., n, $yi\in\{+1,-1\}$, where xi denotes the input data, yi denotes the output labels, n denotes the number of samples, and the hyperplane can be expressed as:

$$f(x) = w \cdot x + b = 0 \tag{1}$$

where w denotes the weight vector, and b is the threshold. The optimization problem of SVM is described as:

$$\min \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \tag{2}$$

$$s.t. \ \ y_i \cdot (w \cdot x_i + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

where C is the penalty parameter, which is used to control the degree of punishment for the error sample.

The above problem can be transformed into Lagrange dual form, and the final decision function is:

$$f(x) = sign(\sum_{i=1}^{n}\alpha_i^* y_i (x_i \cdot x) + b^*) \tag{3}$$

$f(x)$ is the decision function. When different sample data is fed into the function, the decision function will predict the corresponding label.

2.2 Particle Swarm Optimization Algorithm

PSO is an intelligent evolutionary algorithm proposed by Kennedy and Eberhart based on the foraging process of birds [21]. In the PSO algorithm, the process of optimizing the problem is regarded as the process of bird feeding. Each particle in the PSO algorithm is abstracted as a solution to the optimization problem, which is described by two parameters: the position and the velocity of the particle. In each iteration, the velocity and position of the

particle are updated by the formulas:

$$v_{id}(t+1) = wv_{id}(t) + c_1 r_1[pbest_{id}(t) - x_{id}(t)] + c_2 r_2[gbest_d(t) - x_{id}(t)] \quad (4)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (5)$$

where, *i* denotes the *i*th particle; *d* denotes the dimension; *t* is the number of iterations; *xid* denotes the position of the *i*th particle in the *d* dimension, and *xid* is limited in the interval [*popmin*, *popmax*]; *vid*(*t*) denotes the velocity of the particle; *w* is the inertial weight, which controls the influence of contemporary velocity on the next generation's; *pbestid*(*t*) denotes the individual best position, and *gbestd*(*t*) denotes the global best position, c_1 and c_2 are acceleration factors; r_1 and r_2 are the random numbers in interval [0, 1].

Set $z=c_1r_1[pbestid(t)\text{-}xid(t)]+c_2r_2[\ gbestd(t)\text{-}xid(t)]$. Figure 1 shows the above two formulas are represented with vectors in two dimensions. Figure 1(a) denotes the velocity formula, and the velocity *v*(*t*+1) is generated by *wv*(*t*) and *z*. Figure 1(b) denotes the position formula; the position of the new particle *x*(*t*+1) is added by the position *x*(*t*) and the velocity *v*(*t*+1).

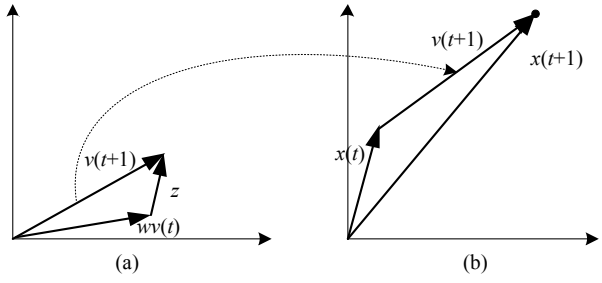

Figure 1: Updating of a PSO algorithm with vector representation in two dimension. (a) velocity update schematic. (b) position update schematic

3 Mutation Strategy and Inertia Weight Strategy of PSO

3.1 Mutation Strategy of PSO

The mutation strategy of PSO is mainly to change the position of the particle. Figure 2 shows the effect of mutation strategy on PSO algorithm. The solid line denotes the original position and the dotted line denotes the new position after the mutation. Since the mutation strategy does not directly affect the velocity of the particle, the vector *v*(*t*+1) in Figure 2(a) does not change. The position of the particle is changed from *x*(*t*) to *x*(*t*)' after mutation in Figure 2(b), and the position of the next generation of particle is changed from *x*(*t*+1) to *x*(*t*+1)' by adding with *v*(*t*+1). If we adjust the particle's moving distance reasonably, we can control particle movement in a certain range, which is beneficial for the particle to jump out of the local optimal solution.

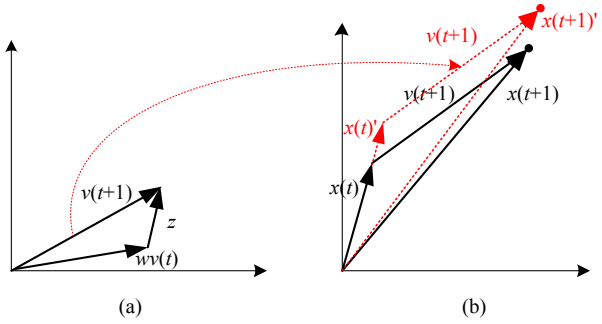

Figure 2: The effect of mutation strategy on the PSO algorithm. (a) speed update schematic; (b) position updating schematic after introducing the mutation strategy

Alireza proposed an adaptive mutation strategy, and the distance of particle moving can be automatically adjusted according to fitness values [53]. The calculation method of variation is as follows:

$$x'_{ij} = x_{ij} + M \times \beta_{ij} \quad (6)$$

$$M_t = pop_{\max} \times \tanh\left[\frac{1}{\alpha} \times F(gbest_d(t))\right] \quad (7)$$

where *xi*' is the position of the *i*th particle after mutation. β*ij* represents a random number that obeys a Gaussian distribution with a mean of 0 and a standard deviation of 1. Parameter *Mt* can control the distance of mutation, which is calculated according to the fitness of global best position. *popmax* is the maximum value of the particles search space, and tanh is a monotonic increasing function. *F*(*gbestd*(*t*)) represents the best fitness of the population. Because the value of tanh is in the interval [0, 1], the value of *Mt* can be controlled adaptively within a certain range.

Stacey and Jancic proposed a Gaussian mutation algorithm (GPSO) [54]. The basic idea is changing the position of a particle when it is in the local optimal solution, and the mutation formula of particle *xi* is:

$$x'_i = x_i(1 + Gaussian(\sigma)) \quad (8)$$

where *xi* is the position of the current particle and *xi*' is the position after mutation. *Gaussian*(σ) is a random number that obeys a Gaussian distribution with a mean of 0 and a standard deviation of σ. The value of σ is 0.1 times the length of search space. The ability of the particle to jump out of the local optimal solution is increased by mutation.

When using Gaussian PSO, a Gaussian distribution operator will be added to the position of a particle. According to the characteristics of Gaussian distributions, the value of the operator is smaller in most cases, resulting in the particle moving in a small range. Therefore, the Gaussian mutation has a strong local search ability and poor global search ability. In addition, there are many variants of the Gaussian strategy, such as Zhan and Lu's neighbor heuristic and Gaussian cloud learning particle swarm optimization algorithm [55].

Wang and Li proposed a Cauchy mutation strategy (CPSO) [56]. The mutation formula of the particle is:

$$p_g^{'}(i) = p_g(i) + W(i) \times N(pop_{\min}, pop_{\max}) \quad (9)$$

$$W(i) = \frac{1}{n}\sum_{j=1}^{n} V[j][i] \quad (10)$$

where W is the mean vector of all particle velocities. n is the size of the population. N is a random number that obeys the Cauchy distribution. Because the probability density function image of a Cauchy distribution is a relatively smooth strip, the two ends are larger and the middle is smaller, which makes the position of the particles change greatly, so the Cauchy mutation has stronger jumping ability. In addition, there are many variants based on Cauchy mutation, such as adding a scaling factor on the Cauchy mutation to control the distance the particle moves [57].

Brockmann and Sokolov found the Levy flight pattern [58], which indicated that most of the cases are changed in a small range, and occasionally a small part of the situation would move to a distant position. Hakl and Uguz applied Levy mutation in particle swarm optimization (LFPSO) [59], and the mutation formula is:

$$x_i^{'} = x_i + \alpha + Levy(\beta) \quad (11)$$

$$x_i^{'} = x_i + random(size(D)) + Levy(\beta) \quad (12)$$

where α denotes the step size of the particle moves, and $Levy(\beta)$ denotes the distribution of the Levy with parameter β. β is a variable between [0, 2]. Due to the occasional long distance movement of Levy flight, the particle's position may move in a wide range occasionally during mutation, causing the particle to jump out of the local optimal solution.

Wang and Wu combined the Gaussian, Cauchy and Levy mutation strategies by an adaptive approach [60]. Nishio and Kushida proposed an adaptive PSO with multi-dimensional mutation strategy [61].

Li and Liu proposed a stochastic mutation method (RPSO) [62]. The particle mutated by the following formula:

$$x_i = x_i + (pop_{\max} - pop_{\min}) \cdot r \quad (13)$$

where *popmax* and *popmin* denote the maximum and the minimum of the particle search range respectively. r is a random number obeying a uniform distribution in the interval (0, 1).

Zhang and Lu proposed a feedback mutation particle swarm optimization algorithm (FBPSO) [63]. The mutation formula is:

$$x_i^{'} = x_i + \beta \cdot x_i \quad (14)$$

$$\beta \sim N(0, \sigma^2) \quad (15)$$

$$\sigma = \sqrt{\left|\frac{fit(i) - fitgbest}{fitavg - fitgbest}\right| + 0.1} \quad (16)$$

where β obeys a Gaussian distribution with the mean of 0 and the standard deviation of σ. $fit(i)$ is the fitness of the ith particle, and *fitgbest* is the global fitness, *fitavg* is the mean fitness of all the particles. Since σ changes according to fitness, this mutation strategy can adjust the distribution of mutation positions based on the information of the particle.

3.2 Inertia Weight Strategy of PSO

Kennedy and Eberhart introduced the inertial weight parameter in the original PSO algorithm, and proposed the particle swarm algorithm with inertial weight [21].

Figure 3 shows the effect of inertia weight in a PSO algorithm. In Figure 3(a), the vector changes from $wv(t)$ to $wv(t)$' after changing the inertia weight w, and then the velocity changes from $v(t+1)$ to $v(t+1)$'. In Figure 3(b), the new position of the particle is changed from $x(t+1)$ to $x(t+1)$' by adding $v(t+1)$' to the position $x(t)$. Therefore, the inertia weight in PSO algorithm essentially changes the position of particles. When using a larger inertia weight,

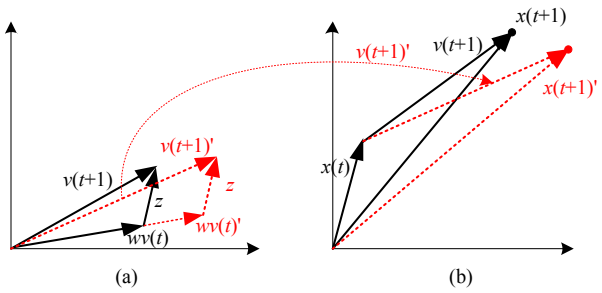

Figure 3: The effect of the inertia weight on PSO algorithm. (a) the speed update schematic after changing the inertia weight. (b) the position update schematic after changing the inertia weight

the position of particle will also change greatly, which facilitates the global exploration of the algorithm in a large surrounding space. When the inertia weight is smaller, the position changes less, which facilitates the local search of the algorithm in the additional small space.

PSO algorithm is nonlinear and highly complex in the search process. In order to improve the ability to monitor the population, Alireza proposed an inertia weight strategy with fitness feedback [53]. The formula is:

$$w(t) = 0.5\left\{1 + \tanh\left[\frac{1}{\alpha} \times F(gbest_d(t))\right]\right\} \tag{17}$$

where tanh and $F(gbestd(t))$ are the same as the definitions in formula (7), which are abbreviation for hyperbolic tangent and the fitness of the current best solution. In this strategy, the value of inertia weight is limited to the interval [0.5, 1]. When the fitness does not decrease, $w(t)$ is large and changes slowly, which benefits to global exploration. When the fitness reduces, $w(t)$ is small, which benefits to local exploitation.

Eberhart and Shi proposed a random inertial weight (RANDPSO) [47]. The inertia weight is calculated as:

$$w(t) = 0.5 + \frac{rand}{2} \tag{18}$$

where *rand* is the random number between [0, 1]. w is a random number in the interval [0.5, 1]. Since the value of inertia weight is random, the result of the algorithm is contingent.

Yang and Gao proposed an exponential inertial weight strategy (LHNPSO) [64]. The inertial weight formula is

$$w(t) = w_{max} - (w_{max} - w_{min})\left(\frac{t}{t_{max}}\right)^{\alpha} \tag{19}$$

where *wmax* and *wmin* are the maximum and minimum of inertia weight respectively. $\alpha=1/\pi^2$. The inertia weight decreases with the number of iterations, which benefits to global exploration and local exploitation of the algorithm.

Nickabadi and Ebadzadeh proposed an adaptive inertia weight, in which the inertia weight is adjusted by the ratio of successful particles (AIWPSO) [65]. The inertia weight formula is:

$$w(t) = (w_{max} - w_{min})Ps(t) + w_{min} \tag{20}$$

$$Ps(t) = \frac{1}{n}\sum_{i=1}^{n} S_i(t) \tag{21}$$

$$S_i(t) = \begin{cases} 1 & ,if\ fit(pbest_i(t)) < fit(pbest_i(t-1)) \\ 0 & ,if\ fit(pbest_i(t)) = fit(pbest_i(t-1)) \end{cases} \tag{22}$$

where, n is the size of the population. $fit(pbesti(t))$ denotes individual fitness. $Ps(t)$ is the search success rate. If the individual fitness of a particle is smaller than the individual fitness of the last generation, the particle searches successfully, and $Si(t)=1$; otherwise, the particle does not search successfully, $Si(t)=0$. The inertia weight is beneficial to supervise the information of particles in the population.

Chauhan and Deep used global best position and individual best position to adjust the inertia weight (DESIWPSO) [66]. The inertia weight of the ith particle is calculated as:

$$w_i(t+1) = \exp\left(-\exp(-R_i(t))\right) \tag{23}$$

$$R_i(t) = \left|gbest(t) - pbest_i(t)\right| \times \frac{t_{\max} - t}{t_{\max}} \tag{24}$$

Taherkhani and Safabakhsh proposed a multi-dimensional changing inertia weight strategy (SAIWPSO) [67]. In this method, the inertia weight of the next generation is automatically adjusted by the position information of the contemporary particle. What's more, the concept of ith particle searches success is introduced:

$$\delta_i(t) = \begin{cases} 1 & ,if\ fit(x_i(\mathrm{t}+1)) < fit(pbest_i(\mathrm{t})) \\ -1 & ,else \end{cases} \tag{25}$$

where $fit(xi(t+1))$ is the fitness of the ith particle in t+1 generation. $fit(pbesti(t))$ is individual fitness. If $fit(xi(t+1)) < fit(pbesti(t))$, the particle searches successfully, and $\delta i(t)=1$, otherwise, the particle does not search successfully, $\delta i(t)=-1$. The inertia weight formula is:

$$w_{ij}(t+1) = \begin{cases} \min\left(1, w_{ij}(t) + (1-w_0) \times \exp\left(\frac{(x_{ij}(t+1) - pbest_{ij}(t))^2}{-2\sigma^2}\right) + \varepsilon\right) & ,if\ \delta_i(t) > 0 \text{ and } \delta_i(t-1) > 0 \\ \max\left(0.1, w_{ij}(t) - w_0 \times \left(1 - \exp\left(\frac{(x_{ij}(t+1) - pbest_{ij}(t))^2}{-2\sigma^2}\right)\right) - \varepsilon\right) & ,if\ \delta_i(t) < 0 \text{ and } \delta_i(t-1) < 0 \\ w_{ij}(t), & \text{else} \end{cases} \tag{26}$$

where $wij(t)$ is the inertia weight of the ith particle in the jth dimension. w_0 is the initial inertia weight. σ is the standard deviation of all particles in the jth dimension. ε is a smaller positive number, ε=0.005.

All of the above inertia weight strategies do not make full use of the information of the particle population, and cannot supervise the status of the population very well. Therefore, we proposed a multi-information fusion inertia weight (MDAPSO) [68]. In order to supervise the state of the population, we refer to other adaptive inertia weights, and introduce the velocity and position of a particle in the

inertia weight, as shown in formulas *Y* and *Z*. In order to control the overall trend of inertia weight, we refer to the decreasing inertia weight with iterations, and introduce λ_1 and λ_2, and in order to avoid the particles falling into local optimal solutions, we introduce a random disturbance. In order to increase the diversity of inertia weight, the inertia weight w_{id} is changed at each iteration, each particle and each dimension. Finally, the formula of inertia weight is as follows:

$$w_{id}(t)=\begin{cases}\min\left(1, w_{id}(t-1)+\left((1-\alpha)\cdot Y+(1-\beta)\cdot Z+\gamma\cdot rand\right)\times\lambda_1(t-1)\right), & case1\\ \max\left(0.1, w_{id}(t-1)-\left(\alpha\cdot Y+\beta\cdot(1-Z)-(1-\gamma)\cdot rand\right)\times\lambda_2(t-1)\right), & case2\\ w_{id}(t-1), & case3\end{cases} \tag{27}$$

$$Y=\left|v_{ij}(t)/v_{max}\right| \tag{28}$$

$$Z=\exp\left(-\left|\frac{x_{ij}(t)-pbest_{ij}(t)}{x_j(t)_{average}-pbest_{ij}(t)}\right|\right) \tag{29}$$

$$x_j(t)_{average}=\frac{1}{n}\sum_{i=1}^{n}x_{ij}(t) \tag{30}$$

$$\lambda_1(t)=0.63\times e^{-\frac{t}{t_{\max}}}+0.56 \tag{31}$$

$$\lambda_2(t)=1.2-0.4\times\left(e^{\frac{t-t_{\max}/2}{t_{\max}}}-0.6\right) \tag{32}$$

$$\begin{cases}case1: & if\ \ \delta_i(t-1)>0 \text{ and } \delta_i(t-2)>0\\ case2: & if\ \ \delta_i(t-1)<0 \text{ and } \delta_i(t-2)<0\\ case3: & else\end{cases} \tag{33}$$

In the formula, min and max are used to limit the inertia weight *wij*(*t*) within the interval [0.1,1]. α=0.9, β=0.55, γ=0.5. *case*1, *case*2 and *case*3 are three conditions; δ*i*(*t*-1) is used to determine whether the particle searches successfully. *case*1 indicates that the particle has searched successfully for two generations, and *case*2 indicates that the particle has not searched successfully for two generations. *case*3 denotes the above two cases are not satisfied.

In the proposed inertia weight formula (27), we refer to α, β and γ as weight coefficients and limit them in the interval [0, 1]. Weight coefficients denote the influence ratios of *Y, Z* and *rand* on the inertia weight. To obtain the weight coefficients more easily and quickly, we use two coefficients to fix one and subsequently test the other one with the step length of 0.1, and the principle is shown in Figure 4. The method fixes β and γ first. Because the range of each weight coefficient lies in the interval [0, 1], the midpoint of the interval is chosen as the fixed value. We fix β=0.5 and γ=0.5 at first and calculate the fitness of each benchmark function when α is different and finally select the α value when the fitness of each benchmark function is minimum. In the same manner, we fix α and γ to obtain β and fix α and β to obtain γ. Figure 5(a), (b) and (c) show the effect of α on the benchmark functions when β and γ are fixed. The horizontal axis is α, which is in the interval [0,1], and the vertical axis is the error between the fitness and theoretical value. The error is small, and the performance of the algorithm is better. From Fig. 5(a), (b) and (c), we note that when α=0.9, the algorithm has a small error. From Fig. 5(d), (e) and (f), we find that β=0.5. From Fig. 5(g), (h) and (i), we observe γ=0.5. We obtain the optimal coefficient of each solution, but the solutions are optimal under certain conditions, and their combination is not necessarily the optimal solution of the algorithm. Therefore, we need to finely adjust α, β and γ to select the most appropriate solution near the three solutions. Eventually, we find that when α=0.9, β=0.55 and γ=0.5, the algorithm achieves a better result.

The pseudo code for our improved inertia weight PSO algorithm is shown in Algorithm 1.

Algorithm 1

Initialize values: *n, D, c1, c2, tmax, popmin, pop max, vmin, vmax, wid*
for *i*=1 to *n*
initialize *xi=xi1, xi2, xi3,..., xiD, vi=vi1,vi 2,vi 3,...,vi D*
end
evaluate the values *fitness*(*xi*)
pbesti=xi
set the particle with best fitness to *gbest*
for *t*=1 to *tmax*
for *i*=1 to *n*
    for *d*=1 to *D*
        vid(*t*+1)=*widvid*(*t*)+*c1r1*[*pbestid*(*t*)-*xid*(*t*)]+*c2r2*[*gbestd*(*t*)-*xid*(*t*)]
    end
    x(*t*+1)= *x*(*t*)+ *v*(*t*+1)
    evaluate the values *fitness*(*xi*)
    if (*fitness*(*xi*)< *fitness*(*pbesti*))
        pbesti=xi
    end
    if (*fitness*(*xi*)< *fitness*(*gbest*))
        gbest=xi
    end
end
for *i*=1 to *n*
    for *d*=1 to *D*
        update w_{id}
end
end
end

The flow chart of the algorithm is shown in the figure 6.

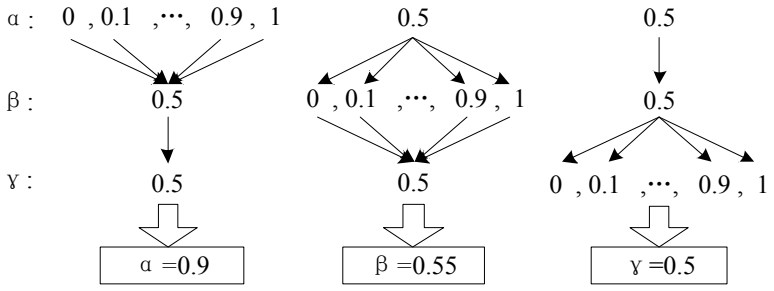

Fig. 4 Method for obtaining parameters α, β and γ in this paper

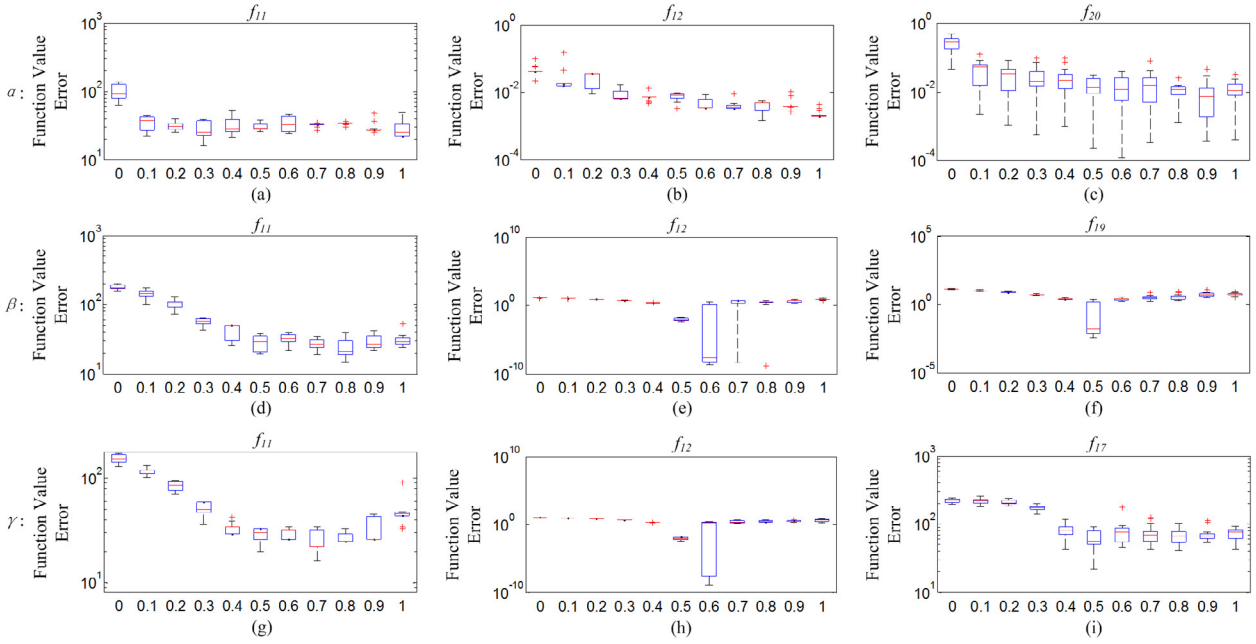

Fig. 5 Effects of different α, β and γ on benchmark functions

4 Comparing the Different Inertia Weight Strategies and Mutation Strategies

4.1 Benchmark Function and Parameter Setting

In order to test the performance of the improved strategies, the different inertial weight strategies and mutation strategies were applied to optimize the benchmark functions that Taherkhani and Safabakhsh [67] used. All the benchmark functions were optimized for the minimum value. Table 3 shows the formula, dimension, search space and global minimum value of each test function. f_1-f_8 are unimodal benchmark functions, mainly to test the optimizing precision f_9-f_{15} are multimodal benchmark functions, and f_{16}-f_{20} are rotating multimodal benchmark functions, mainly to test the ability of the algorithm jumping out of local optimal solutions. The parameters of the PSO algorithm have a great influence on the algorithm. In order to ensure the fairness of testing, all the other parameters of the algorithm were the same except the inertia weight strategy and mutation strategy adopted by different literatures: population size n=20, learning factor c_1=1.49445, c_2=1.49445, maximum velocity v_{max}=0.2(*popmax-popmin*), minimum velocity v_{min}=-v_{max}, dimension D and particle search range [*popmin*,*popmax*] are determined by the benchmark function, iteration number t_{max}=10000D, and each algorithm runs 30 times independently. The final result of each benchmark function is the mean value of 30 results. The experimental platform is an HPZ640 workstation with 32G memory and Windows7 64 bit operating system. The software is MATLAB R2014a and parallel computing is used. We provide an example for parallel computing (https://github.com/zhangming8/APSO). In this example, it is

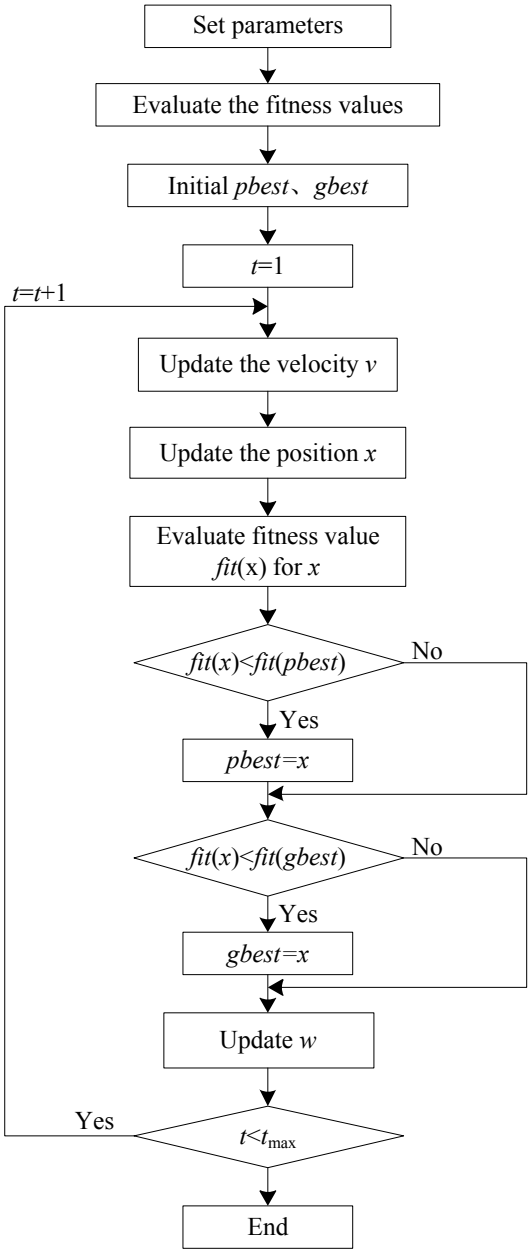

Fig 6. Flow chart of MDAPSO algorithm

the implementation of the improved PSO proposed by Alireza [53], so some of the parameters are the same as in the original paper. Table 1 and Table 2 are the summaries of the improvement strategies that have been discussed earlier.

4.2 Comparing Mutation Strategies without Inertial Weight

The benchmark functions were optimized by PSO with different mutations when the inertia weight strategies were not used. The result is shown in Table 4. The "BestNumber" of the last line in the table denotes the best number of optimization result of each PSO algorithm. As can be seen from Table 4, when the inertia weight strategy is not used, the maximum value of BestNumber corresponding to m_4 is 5, so the performance of mutation m_4 is the best. From the results, the good results of m_4 mainly concentrated in the multimodal benchmark functions. The main reason is that m_4 introduces the Levy flight strategy, which makes the particle occasionally produce a larger jump range and promotes the algorithm to jump out of the local optimal solution. The performance of m_6 and m_3 are also not bad. The main reason is that m_6 used an adaptive mutation strategy in which the mutation distance can be changed according to the fitness value of the population, so as to better monitor the information of the population and promote the algorithm to jump out of the local optimal solution. m_3 used a uniformly distributed mutation operator, and the overall optimization performance is also good.

4.3 Comparing Inertial Weight Strategies without Mutation

In this section, the benchmark functions were optimized by PSO with different inertia weight when the mutation strategies were not used. The result is shown in Table 5. When the mutation strategy is not used, the maximum value of BestNumber corresponding to w_6 is 8, so the performance of w_6 is the best. It shows good performance not only in unimodal benchmark functions, but also in multimodal benchmark functions. The main reason is that w_6 improves the diversity of the population, enhances the supervising ability of the population, and balances the global exploration and local exploitation ability very well. In addition, the performance of w_5 is also not bad, especially in unimodal benchmark functions. The main reason is that w_5 uses the position of the particle to adjust the inertia weight. Meanwhile, the inertia weight changes in each iteration, each particle and each dimension, which helps to improve the diversity of the population.

4.4 Comparing the best inertia weight strategy and mutation strategy

In the previous two sections, the best mutation strategy and the best inertia weight strategy were obtained. We also compared the best inertia weight and the best mutation

Table 1: Formulas of mutation strategies

Name	Author	Mutation strategies	Reference
m_1	Stacey et al.	$x'=x(1+Gaussian(\sigma))$	[54]
m_2	Wang et al.	$p_g'(i)=p_g(i)+W(i)\times N(popmin,popmax)$ $W(i)=\frac{1}{n}\sum_{j=1}^{n} V[j][i]$	[56]
m_3	Li et al.	$x'=x+\alpha+(pop_{max}-pop_{min})\times r$	[62]
m_4	Brockmann et al.	$x'=x+\alpha+Levy(\beta)$ $x'=x+random(size(D))+Levy(\beta)$	[59]
m_5	Zhang et al.	$x'=x+\beta\times x$ $\beta\sim N(0,\sigma^2)$ $\sigma=\sqrt{\left\|\frac{fit(i)-fitgbest}{fitavg-fitgbest}\right\|+0.1}$	[63]
m_6	Alireza et al.	$x'=x+M\times\beta$ $M=pop_{max}\times\tanh[\frac{1}{\alpha}\times F(gbestd(t))]$	[53]

Table 2: Formulas of inertia weight strategies

Name	Author	Inertia weight strategies	Reference
w_1	Eberhart et al.	$w(t)=0.5+\frac{rand}{2}$	[47]
w_2	Yang et al.	$w(t)=w_{max}-(w_{max}-w_{min})\left[\frac{t}{t_{max}}\right]^{\alpha}$	[64]
w_3	Nickabadi et al.	$w(t)=(w_{max}-w_{min})Ps(t)+w_{min}$ $Ps(t)=\frac{1}{n}\sum_{i=1}^{n} S_i(t)$	[65]
w_4	Chauhan et al.	$w(t+1)=exp(-exp(-R_i(t)))$ $R_i(t)=\lvert gbest(t)-pbesti(t)\rvert\times\frac{t_{max}-t}{t_{max}}$	[66]
w_5	Taherkhani et al.	$w_{ij}(t+1)=\begin{cases}\min\{1,wij(t)+(1-w_0)\times N+\varepsilon\}, if\ \delta i(t)>0\ and\ \delta i(t-1)>0\\ \max\{0.1,wij(t)-w0\times(1-N)-\varepsilon\}, if\ \delta i(t)<0\ and\ \delta i(t-1)<0\\ w_{ij}(t)\quad, else\end{cases}$ $N=exp\left(\frac{(x_{ij}(t+1)-pbestij(t))^2}{-2\sigma^2}\right)$	[67]
w_6	Li et al.	$w_{ij}(t)=\begin{cases}\min\{1,w_{ij}(t-1)+((1-\alpha)Y+(1-\beta)Z+\gamma\cdot rand)\lambda(t-1)\}, case1\\ \max\{0.1,w_{ij}(t)-(\alpha Y+\beta(1-Z)-(1-\gamma)\cdot rand)\lambda(t-1)\}, case2\\ w_{ij}(t-1)\quad, case3\end{cases}$	[68]
w_7	Alireza et al.	$w(t)=0.5\left\{1+tanh\left[\frac{1}{\alpha}\times F(gbestd(t))\right]\right\}$	[53]

Table 3: The information of the benchmark functions

Function	Formula	D	Search space	$f(x)_{\min}$
Sphere	$f_1(x)=\sum_{i=1}^{D}x_i^2$	30	$[-100,100]^D$	0
Rotated hyper ellipsoid	$f_2(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^D$	0
Step	$f_3(x)=\sum_{i=1}^{D}(x_i+0.5)^2$	30	$[-100,100]^D$	0
Branin	$f_4(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(x_1)+10-5/4\pi$	2	$-5\le x_1\le 10$ $0\le x_2\le 15$	0
Rosenbrock	$f_5(x)=\sum_{i=1}^{D-1}[100(x_i^2-x_{i+1})^2+(x_i-1)^2]$	30	$[-5,10]^D$	0
McCormick	$f_6(x)=\sin(x_1+x_2)+(x_1-x_2)^2-1.5x_1+2.5x_2+0.9133$	2	$-1.5\le x_1\le 4$ $-3\le x_2\le 4$	0
Beale	$f_7(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$	2	$[-4.5,4.5]^D$	0
Bukin N.6	$f_8(x)=100\sqrt{\lvert x_2-0.01x_1^2\rvert}+0.01\lvert x_1+10\rvert$	2	$-15\le x_1\le -5$ $-3\le x_2\le 3$	0
Schwefel	$f_9(x)=\sum_{i=1}^{D}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)+418.982D$	30	$[-500,500]^D$	0
Rastrigin	$f_{10}(x)=\sum_{i=1}^{D}[x_i^2-10cos(2\pi x_i)+10]$	30	$[-5.12,5.12]^D$	0
Noncontinuois Rastrigin	$f_{11}(x)=\sum_{i=1}^{D}[y_i^2-10cos(2\pi y_i)+10]$ $y_i=\begin{cases}x_i & \lvert x_i\rvert<0.5\\ \frac{round(2x_i)}{2} & \text{else}\end{cases}$	30	$[-5.12,5.12]^D$	0
Ackley	$f_{12}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)+20+e$	30	$[-32,32]^D$	0
Griewank	$f_{13}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	30	$[-600,600]^D$	0
Levy	$f_{14}(x)=\sin^2(\pi y_1)+\sum_{i=1}^{D-1}\left((y_i-1)^2(1+10\sin^2(\pi y_1+1))\right)+(y_D-1)^2(1+\sin^2(2\pi y_D))$ $y_i=1+\frac{x_i-1}{4}$	30	$[-10,10]^D$	0
Shubert	$f_{15}(x)=\left(\sum_{i=1}^{5}i\cos((i+1)x_1+i)\right)\left(\sum_{i=1}^{5}i\cos((i+1)x_2+i)\right)+186.7309$	2	$[-10,10]^D$	0
Rotated Schwefel	$f_{16}(x)=\sum_{i=1}^{D}-y_i\sin\left(\sqrt{\lvert y_i\rvert}\right)+418.982D$ $Y=M\times X$	30	$[-500,500]^D$	0
Rotated Rastrigin	$f_{17}(x)=\sum_{i=1}^{D}[y_i^2-10cos(2\pi y_i)+10]$ $Y=M\times X$	30	$[-5.12,5.12]^D$	0
Rotated Noncontinuous Rastrigin	$f_{18}(x)=\sum_{i=1}^{D}[z_i^2-10cos(2\pi z_i)+10]$ $z_i=\begin{cases}y_i & \lvert y_i\rvert<0.5\\ \frac{round(2y_i)}{2} & \text{else}\end{cases}, Y=M*X$	30	$[-5.12,5.12]^D$	0
Rotated Ackley	$f_{19}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}y_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi y_i\right)+20+e$ $Y=M\times X$	30	$[-32,32]^D$	0
Rotated Griewank	$f_{20}(x)=\frac{1}{4000}\sum_{i=1}^{D}y_i^2-\prod_{i=1}^{D}\cos(\frac{y_i}{\sqrt{i}})+1$ $Y=M\times X$	30	$[-600,600]^D$	0

Table 4: The result of benchmark function (Mean) of PSO mutation strategies without inertia weight (w=1)

	m_1	m_2	m_3	m_4	m_5	m_6
f_1	3.14E+02	8.20E-03	**5.43E-04**	1.43E-02	1.87E+02	5.50E-03
f_2	4.82E-03	1.19E-01	7.59E-02	1.64E-01	2.31E-02	**6.35E-04**
f_3	3.25E+02	5.56E-03	5.47E+01	1.74E-02	**1.60E-03**	6.83E-03
f_4	1.59E-05	1.45E-05	**1.87E-06**	1.42E-05	1.56E-05	9.18E-05
f_5	1.52E-04	1.01E-03	5.45E-04	8.55E-03	1.21E-04	**8.51E-05**
f_6	**8.45E-06**	9.26E-06	8.21E-04	9.19E-06	8.30E-05	4.57E-02
f_7	**3.45E-06**	5.10E-03	8.06E-06	3.69E-05	5.54E-06	2.81E-02
f_8	2.58E-02	2.33E-02	1.96E-02	**3.85E-03**	2.29E-02	4.26E-01
f_9	6.12E+01	2.30E+02	6.92E+00	2.33E+02	5.93E+00	**1.46E-01**
f_{10}	1.61E+01	1.75E-03	2.08E+01	**1.89E-05**	9.21E+00	8.72E-05
f_{11}	1.13E+01	2.15E-03	1.65E+01	**3.25E-05**	2.89E+00	9.42E-04
f_{12}	1.11E+00	**6.72E-04**	1.32E+00	4.06E-03	1.02E-02	9.82E-03
f_{13}	3.56E+00	**6.53E-03**	4.90E+00	6.73E-02	1.44E+00	6.91E-02
f_{14}	5.90E-01	1.42E-05	1.51E+00	**2.04E-06**	7.12E-01	1.50E-05
f_{15}	6.73E-03	7.91E-03	5.98E-03	7.13E-03	**4.98E-03**	9.42E-02
f_{16}	7.00E+02	6.73E+01	**5.07E+00**	6.83E+01	6.19E+02	4.10E+02
f_{17}	1.71E+01	2.47E-03	2.09E+01	**1.13E-04**	1.80E+01	3.68E-03
f_{18}	1.61E+01	2.65E-02	**5.16E-03**	4.23E-02	1.36E+01	9.91E+02
f_{19}	1.08E+00	4.72E-03	1.32E-03	2.41E-03	1.02E+00	**1.25E-03**
f_{20}	3.62E+00	7.84E-03	5.03E+00	6.46E-02	**1.69E-03**	3.57E-03
BestNumber	2	2	4	**5**	3	4

Table 5: The result of benchmark functions (Mean) of PSO inertia weight strategies without mutation

	w_1	w_2	w_3	w_4	w_5	w_6	w_7
f_1	6.71E-98	8.86E+00	3.45E-18	1.11E+01	1.87E-122	**5.59E-161**	6.55E-05
f_2	2.80E-100	1.78E+02	3.38E-49	2.07E+02	2.60E-128	**6.04E-164**	5.18E-05
f_3	4.43E-30	6.30E+00	3.98E-06	1.08E+01	4.91E-29	**1.65E-32**	2.09E-04
f_4	2.13E-05	2.13E-05	2.13E-05	2.13E-05	**2.13E-06**	2.13E-05	1.83E-06
f_5	2.25E+00	4.23E+01	1.87E+00	1.93E+01	1.87E+00	1.27E+00	**1.99E-05**
f_6	1.05E-02	**7.72E-06**	**7.72E-06**	**7.72E-06**	**7.72E-06**	1.05E-02	4.57E-02
f_7	**2.55E-03**	5.09E-03	1.78E-02	1.02E-02	5.09E-03	5.09E-03	2.54E-02
f_8	2.04E-03	2.80E-03	2.48E-03	1.66E-03	**1.44E-03**	2.80E-03	4.26E-01
f_9	4.40E+02	5.80E+02	5.81E+02	4.96E+02	5.80E+02	4.03E+02	**1.31E-01**
f_{10}	6.05E-06	7.19E-06	6.61E+00	6.61E+00	6.85E+00	**5.67E-07**	9.20E-06
f_{11}	3.98E+00	5.30E+00	4.29E+00	5.82E+00	7.61E-03	**1.25E-04**	3.31E-04
f_{12}	1.60E-01	5.64E-01	6.31E-01	5.21E-01	6.28E-01	2.07E-02	**7.74E-04**
f_{13}	2.96E-03	2.83E-01	2.03E-02	6.06E-01	1.45E-02	**7.40E-04**	1.36E-03
f_{14}	7.17E-01	7.83E-01	1.30E+00	2.17E-01	1.27E+00	8.13E-01	**9.76E-05**
f_{15}	**8.85E-07**	**8.85E-07**	**8.85E-07**	**8.85E-07**	**8.85E-07**	8.79E-07	6.70E-02
f_{16}	6.03E+02	5.89E+02	6.66E+02	6.94E+02	6.16E+02	**5.92E+02**	5.19E+03
f_{17}	7.03E-02	7.35E-05	7.76E-04	7.96E-01	7.74E-05	**6.95E-05**	9.44E-05
f_{18}	5.11E+00	7.19E+00	7.59E+00	7.08E+00	**4.70E+00**	4.85E+00	1.04E+02
f_{19}	3.06E-01	6.25E-03	6.38E-01	**6.24E-04**	6.13E-01	1.78E-03	8.69E-04
f_{20}	1.87E-03	1.93E-01	1.26E-02	5.50E-01	**1.92E-05**	1.70E-03	3.27E-05
BestNumber	2	2	2	3	6	**8**	4

strategy together, and the results are shown in Table 6. It can be seen from Table 6 that the overall performance of inertia weight and mutation strategy are similar. As for the single peak functions f_1, f_2, f_3, the inertia weight w_6 obtained higher precision than the mutation m_4, so a good inertia weight strategy has great influence on the precision of the PSO algorithm for the single peak function. For the multimodal functions, the mutation strategy shows better results, so the mutation improves the ability of the algorithm to jump out of the local optimal solution.

4.5 Comparing the Combinations of Inertia Weight Strategies and Mutation Strategies

In the PSO algorithm, the combined inertia weight and mutation strategy can further improve the performance of the algorithm. Figure 7 is the PSO algorithm updating process with the combination of inertia weight and mutation strategy. Figure 7(a) denotes velocity updating, and Figure 7(b) denotes position updating. In Figure 7(a) the vector changes from $wv(t)$ to $wv(t)$' after changing the inertia weight w, and then the velocity changes from $v(t+1)$ to $v(t+1)$'. In Figure 7(b), the position of the particle changes from $x(t)$ to $x(t)$' after mutation, therefore, the new position of the particle is changed from $x(t+1)$ to $x(t+1)$' by adding $v(t+1)$' to the position $x(t)$'. The position of the particle is changed by the combination of inertia weight and mutation operations.

In order to compare the combinatorial performance of inertia weight and mutation strategies, benchmark functions were used to optimize the combinations from w_1 to w_7 and m_1 to m_6. The parameters used were the same as in Section 4.1. In order to clearly reflect the performance of each combination, the "BestNumber" of each combination relative to other combinations was calculated. Figure 8 shows the BestNumber when all the inertia weight strategies and mutation strategies were combined. The transverse axis denotes the different inertia weight strategies and the longitudinal axis is the BestNumber. Each figure denotes 7 combinations. By comparing the 42 combinations, the maximum value of BestNumber is 6, which corresponded to the best combination of w_3+m_5 in Figure 8(e). Although inertia weight w_6 is the best without mutation strategy and mutation m_4 is the best without inertia weight strategy, the combination of them (w_6+m_4) is not the best. The main reason is that the mutation strategy directly changes the position of the particle, whereas the

Table 6: Results of the best inertia weight and mutation strategy

	m_4	w_6
f_1	1.43E-02	**5.59E-161**
f_2	1.64E-01	**6.04E-164**
f_3	1.74E-02	**1.65E-32**
f_4	**1.42E-05**	2.13E-05
f_5	**8.55E-03**	1.27E+00
f_6	**9.19E-06**	1.05E-02
f_7	**3.69E-05**	5.09E-03
f_8	3.85E-03	**2.80E-03**
f_9	**2.33E+02**	4.03E+02
f_{10}	1.89E-05	**5.67E-07**
f_{11}	**3.25E-05**	1.25E-04
f_{12}	**4.06E-03**	2.07E-02
f_{13}	6.73E-02	**7.40E-04**
f_{14}	**2.04E-06**	8.13E-01
f_{15}	7.13E-03	**8.79E-07**
f_{16}	**6.83E+01**	5.92E+02
f_{17}	1.13E-04	**6.95E-05**
f_{18}	**4.23E-02**	4.85E+00
f_{19}	2.41E-03	**1.78E-03**
f_{20}	6.46E-02	**1.70E-03**
BestNumber	10	10

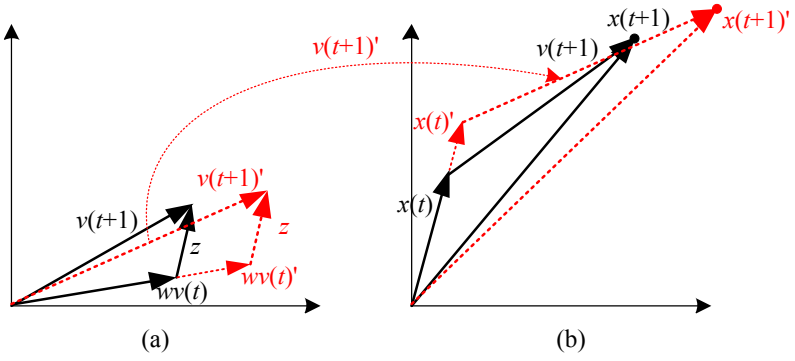

Fig.7 Updating principle of PSO when inertia weight and mutation are used. (a) velocity updates, (b) position updates

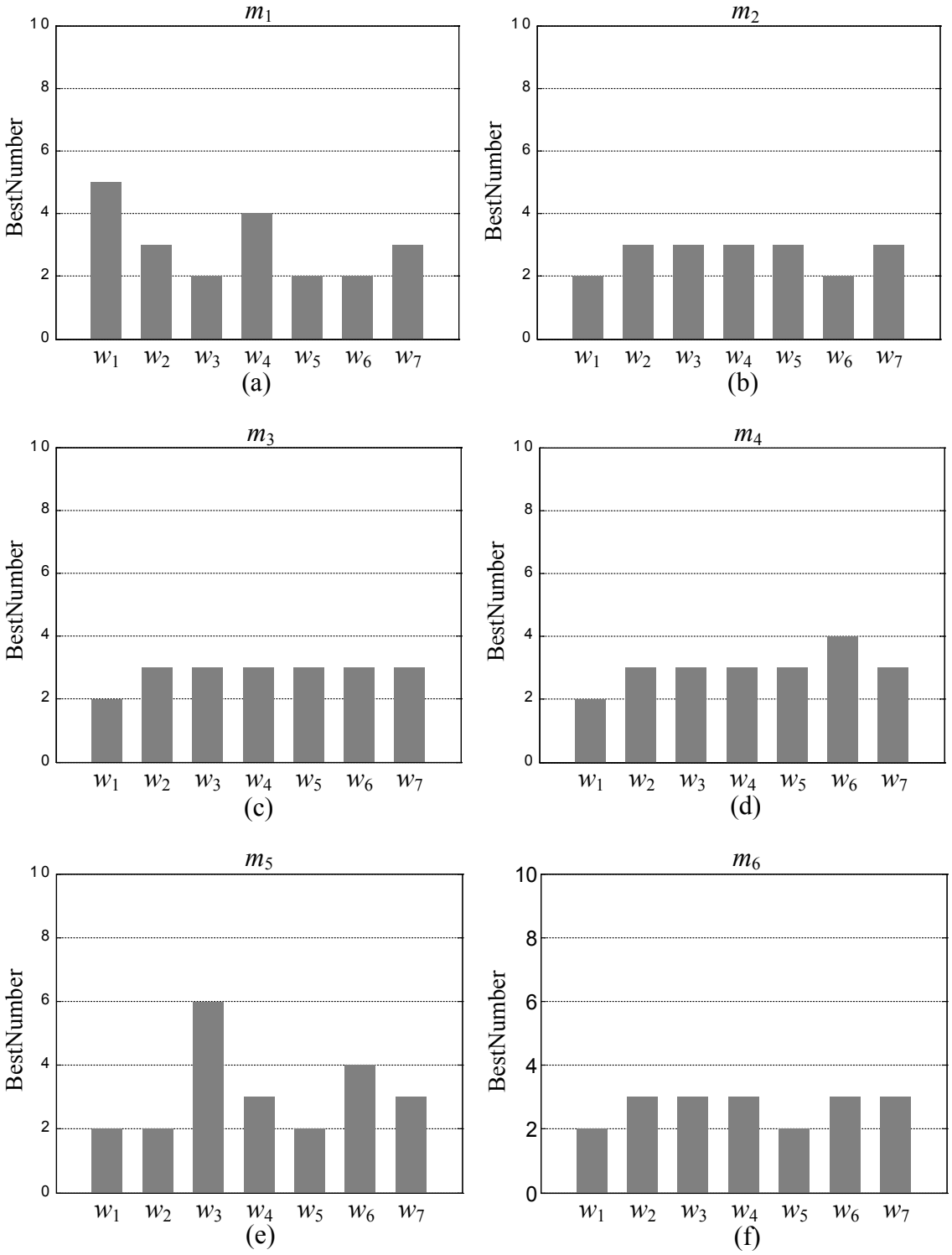

Figure 8: The BestNumber when all the inertia weight strategies and mutation strategies are combined

inertia weight strategy changes the position indirectly by changing the velocity of particle. The essence of both strategies is to change the position of particle. Therefore, using different inertia weight and mutation strategies will interfere with each other and affect the result of the PSO algorithm. The reason for this phenomenon can also be obtained from Figure 7.

5 The Optimization in the Classification of Biomedical Datasets

5.1 Dataset and Test Process

In this section, different classification models were evaluated by the datasets in the UCI machine learning library proposed by Bache and Lichman [69]. Informed consent: Informed consent has been obtained from all individuals included in this study. The datasets used in this study included Breast Cancer, Diabetes, Liver-disorders, Parkinsons, Statlog (heart), and Lung-A (lung cancer) [70], and the specific information of each dataset is given in Table 7.

Table 7: The datasets used in this paper

dataset	number	features	classes
Breast Cancer	683	9	2
Diabetes	768	8	2
liver-disorders	341	6	2
Parkinsons	195	22	2
Lung-A	197	1000	4
Statlog (heart)	270	11	2

In the test process, SVM based on RBF kernel function was used to classify the above datasets. Meanwhile, the PSO algorithm was used to optimize the penalty factor (C) and kernel parameter (g) in SVM, and the process was as follows:

Step 1: Initialize the particle swarm, set the population size and the number of iterations, and initialize the position and velocity of particles randomly;
Step 2: Train the SVM model, and calculate the classification accuracy based on current C and g;
Step 3: Calculate the inertia weight;
Step 4: Update the velocity and position;
Step 5: Update the individual best position *pbest* and global best position *gbest*;
Step 6: Modify the particle position by mutation;
Step 7: Judge the stop condition and return to Step2 if it is not satisfied, otherwise continue;
Step 8: End.

5.2 Results

From Section 4 and Section 5, it is known that m_4 is the best when using mutation strategy, and w_6 is the best when using inertia weight strategy. The combination of w_3 and m_5 is the best when using the combination of inertia weight strategy and mutation strategy. This paper compared the three PSO algorithms based on mutation strategy m_4, inertia weight strategy w_6, and the combination of w_3 and m_5 to optimize penalty factors (C) and RBF kernel parameters (g) in SVM. The other parameters of PSO were set as follows: the population size was 20; the acceleration factors c_1 and c_2 were 1.49445; the maximum number of iteration was 1000; the minimum value of the penalty factor C and the kernel parameter g was 0.01, and the maximum value was 100; random initialization was performed before the algorithm ran. The maximum velocity of the particle was 0.2 times the range of search space, and the minimum velocity was the opposite of the maximum velocity. We used LIBSVM tool proposed by Chang and Lin during the test[71]. 10-fold cross validation was used during the test and repeated 5 times. The test results were averaged. Figure 9 shows the classification accuracy curves of the three PSO algorithms during iteration. Table 8 shows the final accuracies of all the datasets.

From Figure 9, the accuracy of using w_6 on Parkinsons and Lung-A datasets was the highest, the accuracy of using combination w_3+m_5 on other datasets was the highest, and the classification accuracy of using m_4 was the lowest. During the iterations, the combination w_3+m_5 improved rapidly in the early stage on Breast Cancer and Diabetes datasets, and converged to the highest classification accuracy eventually. For the Liver-disorders dataset, the accuracy of w_3+m_5 and w_6 were almost the same at the beginning, but w_3+m_5 converged to the highest accuracy finally. For the Parkinsons dataset, the final accuracies of m_4 and w_6 were the same, and both of them were higher than w_3+m_5. For the Lung-A dataset, all the algorithms converged to a stable value quickly, and the accuracy of w_6 was obviously higher than m_4 and w_3+m_5. For the Statlog (heart) dataset, w_3+m_5 not only converged faster but also converged to a higher accuracy than m_4 and w_6. In general, both the combination w_3+m_5 and the inertial weight w_6 that we proposed were the best, which means that a good algorithm does not show the best results when solving all problems, and this phenomenon also conformed to the "no free lunch" theorem proposed by Wolpert and Macready [72]. Therefore, in the actual classification, we can use the combination w_3+m_5 first. If the improvement is not obvious, we can use w_6 and choose a better algorithm. If the two strategies have the same effect, we can use the strategy that is easier to implement.

Table 8: The average classification accuracies of all the datasets (Mean ± standard deviation)

	Breast Cancer	Diabetes	Liver-disorders	Parkinsons	Lung-A	Statlog (heart)
m_4	98.03±0.08	80.32±0.32	79.48±1.10	**98.26±1.07**	82.51±5.89	85.93±2.14
w_6	98.12±0.07	80.58±0.28	80.35±0.55	**98.26±1.07**	**96.43±0.81**	87.11±0.84
w_3+m_5	**98.18±0.13**	**80.81±0.38**	**80.70±0.86**	97.95±1.35	85.51±2.50	**87.41±0.79**

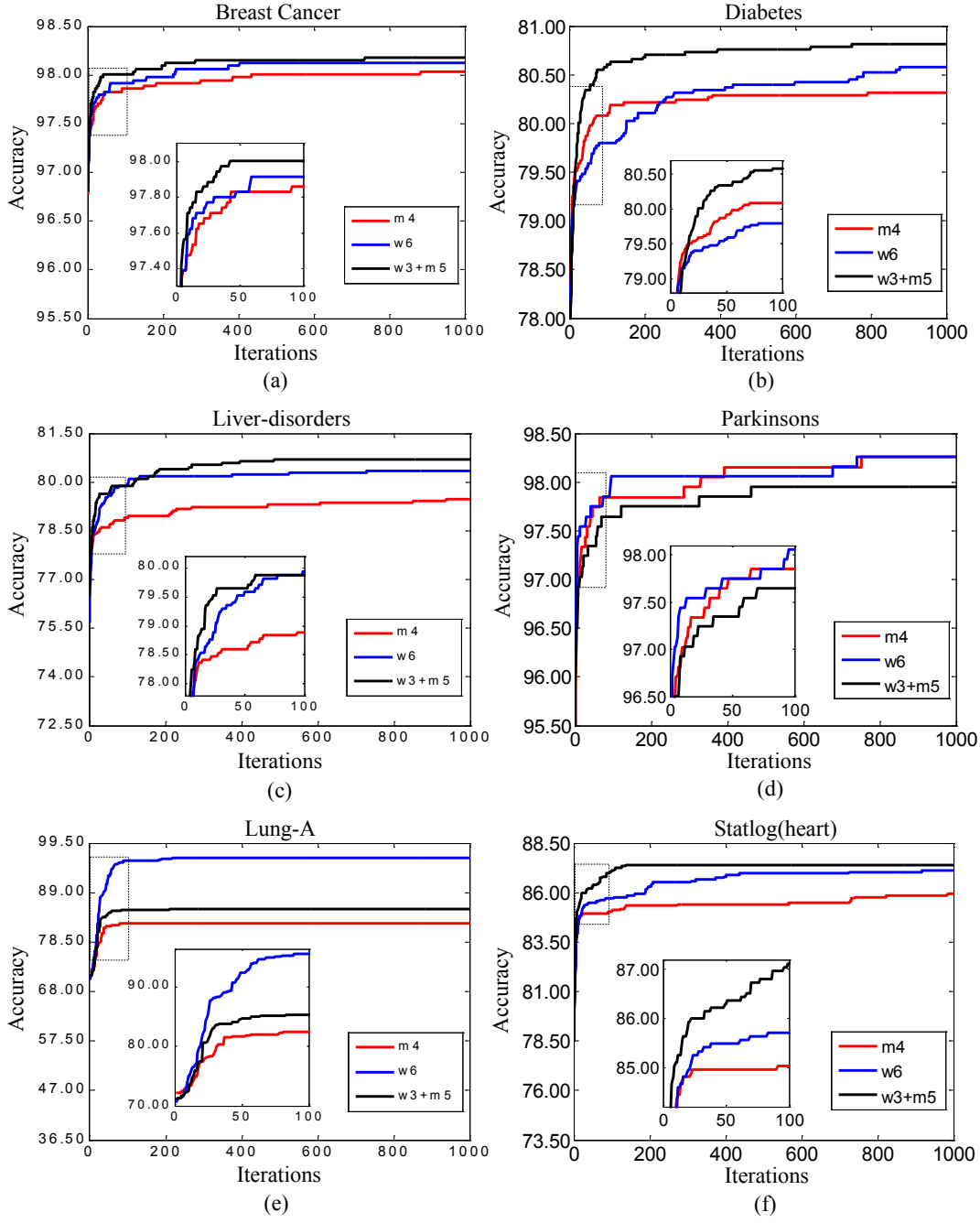

Figure 9: The accuracy curves of the three PSO algorithm during iteration on different datasets

6 Conclusion

This paper compared different inertia weight strategies, mutation strategies and their combinations by optimizing the benchmark function. From the results, we obtained the best mutation strategy without inertia weight, the best inertia weight strategy without mutation, and the best combination of them. At the same time, we found that the effect was not the best when using the best inertial weight and the best mutation strategy simultaneously. The main reason was that the inertia weight and the mutation could interfere with each other. Finally, we used PSO based on different inertia weight and mutation to optimize the penalty factors and kernel parameter of SVM. The classification results showed that the combination of inertia weight and mutation strategy (w_3+m_5) and the inertia weight (w_6) that we proposed had their own advantages on the datasets; both of them could improve the accuracy of biomedical information classification.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (61602017), the National Basic Research Programme of China (2014CB744600), 'Rixin Scientist' Foundation of Beijing University of Technology (2017-RX(1)-03), the Beijing Natural Science Foundation (4164080), the Beijing Outstanding Talent Training Foundation (20140000020124G039), the National Natural Science Foundation of China (61420106005), and the International Science & Technology Cooperation Program of China (2013DFA32180).

Conflict of interest: Authors state no conflict of interest.

References

[1] Joachims T. Making large-scale SVM Learning Practical. Technische Universität Dortmund. 1998:499-526.
[2] Quinlan JR. Induction on decision tree. Machine Learning, 1986;1(1):81-106.
[3] Fukunage K, Narendra PM. A Branch and Bound Algorithm for Computing k-Nearest Neighbors. IEEE Trans Comput, 1975;24(7):750-753.
[4] Bataineh MA, Al-Qudah Z. A novel gene identification algorithm with Bayesian classification. Biomedical Signal Processing and Control, 2017;31:6-15.
[5] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015;521(7553):436.
[6] Chen Z, Huang L, Shen Y, Wang J, Zhao R, Dai J. A new algorithm for classification of ictal and pre-ictal epilepsy ECoG using MI and SVM. International Conference on Signals and Systems. IEEE, 2017:212-216.
[7] Filho AODC, Paiva ACD, Gattass M. Classification of breast regions as mass and non-mass based on digital mammograms using taxonomic indexes and SVM. Computers in Biology & Medicine, 2015;57:42-53.
[8] Zhou WG, Cui YF, Li Y. A modified method combined with a support vector machine and Bayesian algorithms in biological information. International Journal Bioautomotion, 2015;19(2):135-146.
[9] Qu AP, Chen JM, Wang LW, Yuan JP, Yang F, Xiang QM, et al. Segmentation of Hematoxylin-Eosin stained breast cancer histopathological images based on pixel-wise SVM classifier. Science China Information Sciences, 2015;58(9):92105-092105.
[10] Mishra BK, Lakkadwala P, Shrivastava NK. Novel Approach to Predict Cardiovascular Disease Using Incremental SVM. 2013:55-59.
[11] Liu X, Li C, Zhang L, Shi X, Wu S, et al. Personalized Identification of Differentially Expressed Modules in Osteosarcoma. Medical Science Monitor International Medical Journal of Experimental & Clinical Research, 2017;23:774-779.
[12] Parikh KS, Shah TP. Support Vector Machine-A Large Margin Classifier to Diagnose Skin Illnesses. Procedia Technology, 2016;23:369-375.
[13] Fung G, Mangasarian O. Proximal Support Vector Machines. 2001.
[14] Lin CF, Wang SD. Fuzzy support vector machines. IEEE Trans Neural Netw, 2002;13(2):464-471.
[15] Jayadeva, Khemchandani R, Chandra S. Twin Support Vector Machines for Pattern Classification. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2007;29(5):905-10.
[16] Kuo BC, Ho HH, Li CH, Hung CC, Taur JS, et al. A Kernel-Based Feature Selection Method for SVM With RBF Kernel for Hyperspectral Image Classification. IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing, 2013;7(1):317-326.
[17] Prabin A, Veerappan J. Modified Micro Structure Descriptors and Hybrid-RBF Kernel SVM Based Diagnosis of Brain Tumor in MRI Images. Journal of Medical Imaging & Health Informatics, 2015;5(6):1194-1200.
[18] Bousseta R, Tayeb S, Ouakouak IE, Gharbi M, Regragui F. EEG efficient classification of imagined hand movement using RBF kernel SVM. International Conference on Intelligent Systems: Theories and Applications. IEEE, 2016:1-6.
[19] Keerthi S, Lin C. Asymptotic behaviors of support vector machines with gaussian kernel. Neural Computation, 2003;15(7):1667.
[20] Chapelle O, Vapnik V, Bousquet O, Mukerjee S. Choosing Multiple Parameters for Support Vector Machines. Machine Learning, 2002;46(1-3):131-159.
[21] Kennedy J, Eberhart R. Particle swarm optimization. IEEE International Conference on Neural Networks, 1995. Proceedings. IEEE, 2002(4):1942-1948.
[22] Goldberg DE, Goldberg DM, Goldberg DE, Goldberg D, Goldberg ED. Genetic algorithm is search optimization and machine learning. 1989,3(7):2104–2116.
[23] Qinghong WU, Zhang JH, Xin He XU. An ant colony algorithm with mutation features. Journal of Computer Research & Development, 1999,10:014.
[24] Selim SZ, Alsultan K. A simulated annealing algorithm for the clustering problem. Pattern recognition, 1991 Jan 1, 24(10): 1003-1008.

[25] Eskandar H, Sadollah A, Bahreininejad A, Hamdi M. Water cycle algorithm-A novel metaheuristic optimization method for solving constrained engineering optimization problems. Computers & Structures, 2012;110-111(10):151-166.
[26] Pahnehkolaei SMA, Alfi A, Sadollah A, Kim JH. Gradient-based water cycle algorithm with evaporation rate applied to chaos suppression. Applied Soft Computing, 2017;53:420-440.
[27] Gonçalves MS, Lopez RH, Miguel LFF. Search group algorithm: a new metaheuristic method for the optimization of truss structures. Computers & Structures, 2015;153:165-184.
[28] Seyedeh FHN, Alireza A. Adaptive parameter control of search group algorithm using fuzzy logic applied to networked control systems. Soft Computing, 2017(4):1-22.
[29] Zong WG, Kim JH, Loganathan GV. A New Heuristic Optimization Algorithm: Harmony Search. Simulation Transactions of the Society for Modeling & Simulation International, 2016;76(2):60-68.
[30] Ameli K, Alfi A, Aghaebrahimi M. A fuzzy discrete harmony search algorithm applied to annual cost reduction in radial distribution systems. Engineering Optimization, 2015;48(9):1529-1549.
[31] Zhang YD, Zhang Y, Lv YD, Hou XX, Liu FY. Alcoholism detection by medical robots based on Hu moment invariants and predator–prey adaptive-inertia chaotic particle swarm optimization. Computers & Electrical Engineering, 2017.
[32] Wang S, Phillips P, Yang J, Sun P, Zhang Y. Magnetic resonance brain classification by a novel binary particle swarm optimization with mutation and time-varying acceleration coefficients. Biomedizinische Technik/biomedical Engineering, 2016;61(4):431-441.
[33] Zhang Y, Wang S, Ji G. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathematical Problems in Engineering, 2015(1):1-38.
[34] Fernandez M, Caballero J, Fernandez L, Sarai A. Genetic algorithm optimization in drug design QSAR: Bayesian-regularized genetic neural networks (BRGNN) and genetic algorithm-optimized support vectors machines (GA-SVM). Molecular Diversity, 2011;15(1):269-289.
[35] Behravan I, Dehghantanha O, Zahiri SH. An optimal SVM with feature selection using multi-objective PSO. Swarm Intelligence and Evolutionary Computation. IEEE, 2016:76-81.
[36] Kuang F, Zhang S, Jin Z, Xu W. A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection. Soft Computing, 2015;19(5):1187-1199.
[37] Subasi A. Classification of EMG signals using PSO optimized SVM for diagnosis of neuromuscular disorders. Computers in Biology & Medicine, 2013;43(5):576-586.
[38] Ren Y, Hu F, Miao H. The optimization of kernel function and its parameters for SVM in well-logging. International Conference on Service Systems and Service Management. IEEE, 2016:1-5.
[39] Zhang S. Research and Application of Hybrid PSO-BP Neural Network In fracture acidizing well production prediction. Revista de la Facultad de Ingeniería, 2016, 31(6).
[40] Tanweer MR, Auditya R, Suresh S, Sundararajan N, Srikanth N. Directionally Driven Self-Regulating Particle Swarm Optimization algorithm. Swarm & Evolutionary Computation, 2016;28:98-116.
[41] Meng A, Li Z, Yin H, Chen S, Guo Z. Accelerating particle swarm optimization using crisscross search. Information Sciences An International Journal, 2016;329(C):52-72.
[42] Wang CF, Liu K. A Novel Particle Swarm Optimization Algorithm for Global Optimization. Computational Intelligence & Neuroscience, 2016;38(2-3):116-133.
[43] Chen D, Zhang R, Yao C, Zhao Z. Dynamic topology multi force particle swarm optimization algorithm and its application. Chinese Journal of Mechanical Engineering, 2016;29(1):124-135.
[44] Liang H T, Kang F H. Adaptive mutation particle swarm algorithm with dynamic nonlinear changed inertia weight. Optik-International Journal for Light and Electron Optics, 2016, 127(19):8036-8042.
[45] Shi Y, Eberhart R. Modified particle swarm optimizer. Proc. of IEEE ICEC conference, Anchorage. 1998:69-73.
[46] Alireza A, Modares H. System identification and control using adaptive particle swarm optimization. Applied Mathematical Modelling, 2011;35(3):1210-1221.
[47] Eberhart RC, Shi Y. Tracking and optimizing dynamic systems with particle swarms. Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. IEEE, 2001(1):94-100.
[48] Malik RF, Rahman TA, Hashim SZM, Ngah R. New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight. International Journal of Computer Science & Security, 2007;1(2):43-52.
[49] Gholamian M, Meybodi MR. Enhanced comprehensive learning cooperative particle swarm optimization with fuzzy inertia weight (ECLCFPSO-IW). Ai & Robotics. IEEE, 2015:1-7.
[50] Javad AM, Mousa S, Hossein S M. A Novel Flexible Inertia Weight Particle Swarm Optimization Algorithm. Plos One, 2016;11(8):e0161558.
[51] Alireza A, Fateh MM. Identification of nonlinear systems using modified particle swarm optimisation: a hydraulic suspension system. Vehicle System Dynamics, 2011;49(6):871-887.
[52] Cherkassky V. The Nature Of Statistical Learning Theory. Technometrics, 1997;8(6):1564.
[53] Alireza A. PSO with Adaptive Mutation and Inertia Weight and Its Application in Parameter Estimation of Dynamic Systems. Acta Automatica Sinica, 2011;37(5):541-549.
[54] Stacey A, Jancic M, Grundy I. Particle swarm optimization with mutation. Evolutionary Computation, 2003. CEC'03. The 2003 Congress on. IEEE, 2003;2:1425-1430.
[55] Zhan D, Lu H, Hao W, Jin D. Improving particle swarm optimization: Using neighbor heuristic and Gaussian cloud learning. Intelligent Data Analysis, 2016;20(1):167-182.
[56] Wang H, Li C, Liu Y, Zeng S. A hybrid particle swarm algorithm with Cauchy mutation. Swarm Intelligence Symposium. SIS 2007. IEEE. 2007;356-360.
[57] Zhang L, Tang Y, Hua C, Guan X. A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. Applied Soft Computing Journal, 2015;28(C):138-149.
[58] Brockmann D, Sokolov IM. Levy flights in external force fields: from models to equations. Chemical Physics, 2002;284(1–2):409-421.
[59] Haklı H, Uğuz H. A novel particle swarm optimization algorithm with Levy flight [J]. Applied Soft Computing, 2014;23:333-345.
[60] Wang H, Wang W, Wu Z. Particle swarm optimization with adaptive mutation for multimodal optimization. Applied Mathematics and Computation, 2013;221:296-305.

[61] Nishio T, Kushida J, Hara A, Takahama T. Adaptive particle swarm optimization with multi-dimensional mutation. IEEE, International Workshop on Computational Intelligence and Applications. IEEE, 2016:131-136.
[62] Li M, Liu X, Wang X, Lu S, Zhong N. Biomedical classification application and parameters optimization of mixed kernel SVM based on the information entropy particle swarm optimization. Computer Assisted Surgery, 2016;21(1):132-141.
[63] Zhang M, Lu S, Li M, Zhai Q, Zhou J, Lu XF, et al. SVM classification model in depression recognition based on mutation PSO parameter optimization. 2017;8:01037.
[64] Yang C, Gao W, Liu N, Song C. Low-discrepancy sequence initialized particle swarm optimizationalgorithm with high-order nonlinear time-varying inertia weight. Applied Soft Computing Journal, 2015;29:386-394.
[65] Nickabadi A, Ebadzadeh MM, Safabakhsh R. A novel particle swarm optimization algorithm with adaptive inertia weight. Applied Soft Computing, 2011 Jun 1; 11(4):3658-3670.
[66] Chauhan P, Deep K, Pant M. Novel inertia weight strategies for particle swarm optimization. Memetic Computing, 2013;5(3):229-251.
[67] Taherkhani M, Safabakhsh R. A novel stability-based adaptive inertia weight for particle swarm optimization. Applied Soft Computing, 2016;38:281-295.
[68] Li M, Zhang M, Lu SF, Liu XW, Chen H, A multi-information fusion “time-varying triple variable” inertia weight PSO algorithm. Applied Soft Computing, 2018(under review).
[69] Bache K.,Lichman M., UCI Machine Learning Repository. 2013.
[70] Monti S, Tamayo P, Mesirov J, Golub T. Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. Machine Learning, 2003, 52(1-2):91-118.
[71] Chang CC, Lin CJ. LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2011;2(3):27.
[72] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1997;1(1):67-82.