Journal of Classification (2019) 36:576–598
https://doi.org/10.1007/s00357-018-9299-1

Quantum-Behaved Particle Swarm Optimization for Parameter Optimization of Support Vector Machine

Alaa Tharwat[1] · Aboul Ella Hassanien[2,3]

Published online: 21 January 2019

Abstract
Support vector machine (SVM) parameters such as penalty parameter and kernel parameters have a great influence on the complexity and accuracy of SVM model. In this paper, quantum-behaved particle swarm optimization (QPSO) has been employed to optimize the parameters of SVM, so that the classification error can be reduced. To evaluate the proposed model (QPSO-SVM), the experiment adopted seven standard classification datasets which are obtained from UCI machine learning data repository. For verification, the results of the QPSO-SVM algorithm are compared with the standard PSO, and genetic algorithm (GA) which is one of the well-known optimization algorithms. Moreover, the results of QPSO are compared with the grid search, which is a conventional method of searching parameter values. The experimental results demonstrated that the proposed model is capable to find the optimal values of the SVM parameters. The results also showed lower classification error rates compared with standard PSO and GA algorithms.

Keywords Quantum particle swarm optimization (QPSO) · Optimization algorithms · Support vector machine (SVM) · Classification · Parameter optimization

1 Introduction

Support vector machine (SVM) is one of the well-known machine learning methods for classification and regression problems. It was introduced by Vapnik, and it has been used in many applications such as biometrics (Tharwat et al. 2018), bioinformatics (Byvatov and Schneider 2002), and chemoinformatics (Tharwat and Moemen 2017). In SVM classifier, the training patterns are used to train the classification model, and this model is used to classify an unknown pattern. SVM has two main parameters: the penalty parameter (C) and kernel parameters. The penalty parameter determines the trade-off between maximizing the

✉ Alaa Tharwat
engalaatharwat@hotmail.com

1 Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences, 60318 Frankfurt am Main, Germany

2 Faculty of Computers and Information, Cairo University, Giza, Egypt

3 Scientific Research Group in Egypt (SRGE), Giza, Egypt

classification margin and minimizing the training error, while the kernel parameters are used to transform the data from input feature space to a high-dimensional space. Thus, choosing the values of the two parameters controls the performance of SVM.

Swarm intelligence algorithms achieved competitive results when solving optimization problems including parameter tuning problems (Wang and Guo 2013; Yang 2014). Particle swarm optimization (PSO) is one of the well-known optimization algorithms. It is a population-based stochastic optimization algorithm, and it has a small number of parameters; hence, it can be easily implemented. In the PSO algorithm, the particles moved in the search space with a velocity constantly updated by (1) the particle's own experience;and (2) the experience of the whole swarm (Eberhart and Kennedy 1995).

PSO has been applied in many optimization problems. PSO is widely used in image processing. For example, Maitra et al. used the PSO algorithm to search for the optimal thresholding value in image segmentation (Maitra and Chatterjee 2008). Moreover, Akhilesh et al. used a new variant of the PSO algorithm for image segmentation (Chander et al. 2011). In power applications, the PSO algorithm was used for different purposes. Miyatake et al. used the PSO algorithm to search for the maximum power point tracking of multiple photovoltaic and they found that the PSO algorithm converged rapidly to the global maximum power point (Miyatake et al. 2011). Sidhartha Pandan and Narayana Prasad Padhy used PSO to find the optimal location of the static synchronous compensator and its coordinated design with power system stabilizers (Panda and Padhy 2008). They found that the PSO algorithm improved the stability of the power system. The PSO algorithm has been used to solve mathematical problems. Liang et al. used the PSO algorithm to find the global solution of multimodal functions (Liang et al. 2006). Moreover, Zhao Xinchao proposed a perturbed PSO algorithm (Xinchao 2010). He used the new algorithm to search for the optimal solutions for 12 functions, and he found that the proposed algorithm achieved results better than the standard PSO algorithm. In machine learning, PSO was used to search for SVM parameters (Subasi 2013). Moreover, the PSO algorithm was used to adjust the weights of back-propagation (BP) neural networks, and it achieved good results compared with the BP algorithm (Bashir and El-Hawary 2009). In addition, PSO was used for feature subset selection (Lin et al. 2008). In clustering, PSO was used to search for the centroids of a user specified number of clusters, and the PSO algorithm achieved results better than K-means clustering (Merwe et al. 2003). However, due to the stochastic nature of PSO, swarm intelligence algorithms are never guaranteed to find an optimal solution for any problem, but they will often find a good solution if one exists.

Many variants of PSO algorithm were proposed to generate random numbers. Richer and Blackwell used the Levy distribution to improve the performance of the PSO algorithm (Richer and Blackwell 2006). Moreover, Kennedy used a double-exponential distribution with other versions of Gaussian distributions in the PSO algorithm to achieve good results (Kennedy 2004, 2005). The influence of different probability distributions, such as Gaussian, Cauchy, and the exponential probability distributions, was studied by Krohling and Coelho, and they found that these distributions could improve the performance of the PSO algorithm (Krohling 2004; Santos Coelho and Krohling 2005; Krohling and Santos Coelho 2006). Chaos theory was proposed to adapt the inertia weight factor of PSO (Liu et al. 2005). The proposed Chaotic PSO efficiently balances the exploration and exploitation phases (Tharwat and Hassanien 2018).

Recently, Sun et al. proposed a new strategy based on quantum mechanics (Clerc and Kennedy 2002). They used the quantum model to sample around the previous best positions, and then the mean best solution was introduced in the proposed version of PSO, quantum-behaved particle swarm optimization (QPSO) (Sun and Feng 2004; Sun et al. 2004; 2005).

In the QPSO algorithm, the iterative equations are different from that of PSO. Moreover, the QPSO does not require velocity vectors to move the particles as in the PSO algorithm, and the number of adjusting parameters of QPSO was less than the standard PSO; hence, the QPSO algorithm is easier to implement than PSO. The QPSO algorithm has been used to solve different optimization problems and it achieved competitive results (Mikki and Kishk 2006; Li et al. 2007; Omkar et al. 2009).

The main objective of this paper is to use the QPSO algorithm to present a novel QPSO-SVM model for parameters optimization of SVM, which improves the classification performance. This novel approach adjusts the parameters' values to make the optimal separating hyperplane obtainable in linear and nonlinear classification problems.

The rest of the paper is organized as follows: Section 2 presents the related work. In Section 3, the background of the SVM classifier, PSO, and QPSO are introduced; the proposed model (QPSO-SVM) is introduced in Section 4; experimental results and discussion are presented in Section 5; concluding remarks and future work are provided in Section 6.

2 Related Work

There are many studies of tuning SVM parameters empirically by trying a finite number of values and selecting the values that achieve the minimum classification error. This procedure needs an exhaustive search to find the feasible solution, and this is computationally infeasible (Friedrichs and Igel 2005). A grid search algorithm is used to find optimal parameters of SVM. In this algorithm, the parameters vary with a fixed step-size through the parameter space, and the values of each parameter combination are calculated. This algorithm is suitable for adjustment of very few parameters, but it is time-consuming (Chapelle et al. 2002; Wang 2005; Friedrichs and Igel 2005). A leave-one-out (LOO) error and its gradient method were proposed to optimize SVM parameters (Friedrichs and Igel 2005). Different studies used evolutionary algorithms to find the optimal values of SVM parameters to improve the accuracy and stability of the classification model. Shawkat et al. showed that selecting the optimal degree of a polynomial kernel plays a critical role in SVM to ensure a good generalization (Ali and Smith 2003). In another research, PSO was used for parameter determination of SVM and feature selection (Lin et al. 2008). Wu et al. used a hybrid genetic algorithm (GA) for SVM parameter optimization to predict the maximum electrical daily load (Wu et al. 2009). Moreover, ant colony optimization (ACO) was used to search for the optimal SVM parameters (Zhang and Chen 2010). Subasi used PSO algorithm to find SVM parameters for diagnosis of neuromuscular disorders in EMG signals (Subasi 2013). Social emotional optimization (SMO) was used to optimize SVM parameters, and it achieved competitive results (Zhang and Zhang 2015). Bat, dragonfly, and whale algorithms were used also to optimize SVM parameters (Tharwat et al. 2016; Tharwat et al. 2017; Tharwat and Moemen 2017).

3 Preliminaries

3.1 Support Vector Machine Classifier

SVM is one of the most widely used learning algorithms. The goal of SVM is to separate different classes using hyperplanes. The performance of SVM is highly affected by nonlinearly separable data. However, this problem can be solved by transforming the data from the input space to a new higher dimensional space using one of the kernel functions. The data after this transformation can be separated easily. Selecting a suitable kernel function and adjusting its parameters helps the SVM classifier to create linear decisions through a nonlinear transformation. Computationally, searching for the best decision plane is an optimization problem (Chapelle et al. 2002).

Assume we have N training patterns ($X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$), where $\mathbf{x}_i$ is the ith training pattern and each pattern has d attributes and it is in one of two classes $y_i \in \{\pm 1\}$. Thus, the training set is denoted by $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, where $y_1, y_2, \ldots, y_N$ indicate the class labels for $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. In linearly separable data, the line $\mathbf{w}^T\mathbf{x}+b = 0$ represents the decision boundary between the two classes, *positive* and *negative* classes, where $\mathbf{w}$ is a weight vector, b is the threshold or bias, and $\mathbf{x}$ is the input vector or the training pattern (Tharwat 2016a; Tharwat et al. 2017). This line in the two-dimensional space, i.e., $\mathbf{w}^T\mathbf{x} + b = 0$, is represented by a hyperplane in higher dimensional space, and it divides the space into two spaces, namely positive half space (where the patterns from the positive class are located) and negative half space (where the patterns from the negative class are located) (Wang 2005).

The goal of the SVM classifier is to select the values of $\mathbf{w}$ and b to orientate the hyperplane to be as far as possible from the closest patterns and to construct the two planes, $H_1 \rightarrow \mathbf{w}^T\mathbf{x}_i + b = +1$ for $y_i = +1$ and $H_2 \rightarrow \mathbf{w}^T\mathbf{x}_i + b = -1$ for $y_i = -1$, where $\mathbf{w}^T\mathbf{x}_i + b \geq +1$ for positive class and $\mathbf{w}^T\mathbf{x}_i + b \leq -1$ for negative class. These two planes can be combined as follows, $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \; \forall i = 1, 2, \ldots, N$.

The distance from H_1 and H_2 to the hyperplane is denoted by d_1 and d_2, respectively, and these distances represent the margin of SVM. The hyperplane is equidistant from the two planes; hence, $d_1 = d_2 = \frac{1}{\|\mathbf{w}\|}$ and the margin represents the total of d_1 and d_2. The margin width needs to be maximized as follows:

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \;\; \forall i = 1, 2, \ldots, N \end{aligned} \tag{1}$$

Equation 1 represents quadratic programming problem, where $\min \frac{1}{2}\|\mathbf{w}\|^2$ is the objective function and $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0$ represents the constraint. This equation can be formalized into Lagrange formula as follows:

$$\begin{aligned} \min \; L_P &= \frac{\|\mathbf{w}\|^2}{2} - \sum_i \alpha_i (y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) \\ &= \frac{\|\mathbf{w}\|^2}{2} - \sum_i \alpha_i y_i (\mathbf{w}^T\mathbf{x}_i + b) + \sum_{i=1}^{N} \alpha_i \end{aligned} \tag{2}$$

where $\alpha_i \geq 0$, $i = 1, 2, \ldots, N$ represent the Lagrange multipliers and each Lagrange multiplier (α_i) corresponds to one training pattern and L_P represents the primal problem. The values of $\mathbf{w}$, b, and α which can be calculated by differentiating L_P with respect to $\mathbf{w}$ and b and setting the derivatives to zero as follows:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{3}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4}$$

Substituting Eqs. 3 and 4 into Eq. 2, the dual problem can be written as follows:

$$\max\ L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$
$$\text{s.t. } \alpha_i \geq 0,\ \sum_{i=1}^{N} \alpha_i y_i = 0\ \forall i = 1, 2, \ldots, N \tag{5}$$

where L_D represents the dual form of L_P. In dual SVM, the objective function is maximized with respect to α_i, instead of minimizing it with respect to $\mathbf{w}$ and b as in primal SVM as denoted in Eqs. 2 and 5.

The values of $\mathbf{w}$, b, and α are determined by solving Eqs. 3, 4, and 5. The nonzero α's are corresponding to support vectors (SVs), which are the patterns that are closest to the separating hyperplane; thus, SVs achieved the maximum width margin.

In the case of nonseparable data, more misclassified patterns result. Therefore, a slack variable, ϵ_i, is added to relax the constraints of linear SVM, where ϵ_i indicates the distance between the ith training pattern and the corresponding margin hyperplane, and it should be minimized. The objective function of SVM after adding ϵ_i will be as follows:

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \epsilon_i$$
$$\text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \epsilon_i \geq 0\ \ \forall i = 1, 2, \ldots, N \tag{6}$$

where C represents the penalty parameter, and it controls the trade-off between the slack variable penalty and the size of the SVM margin. Lagrange formula of Eq. 6 is as follows:

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \epsilon_i - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \epsilon_i] \tag{7}$$

where $\alpha_i \geq 0$.

The value of ϵ_i is calculated by differentiating L_P with respect to ϵ_i as follows:

$$\frac{\partial L_P}{\partial \epsilon_i} = 0 \Rightarrow C = \alpha_i + \epsilon_i \tag{8}$$

As denoted in Eq. 8, α_i is limited by the upper-bound C, and SVs with $\alpha_i = C$ lie outside the margin or on the margin boundary.

In case of nonlinearly separable data, SVM uses kernel functions to transform the data into a higher dimensional space using a nonlinear kernel function (ϕ) where the data can be linearly separable. The objective function of SVM will be as follows:

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \epsilon_i$$
$$\text{s.t. } y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) - 1 + \epsilon_i \geq 0\ \ \forall i = 1, 2, \ldots, N \tag{9}$$

There are different kernel functions such as follows:

- Linear kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$;
- Radial basis function (RBF) or Gaussian kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$;
- Polynomial kernel of degree d, $K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$ (Scholköpf and Smola 2001).

In this study, the RBF kernel function is applied in the SVM classifier to obtain the optimal or near optimal solutions.

3.2 SVM Parameter Optimization: Illustrative Example

In SVM, the penalty parameter C and kernel function parameters have considerable influence on the classification performance. The parameter C controls the trade-off between the minimization of classification error and maximization of margin. In the RBF kernel function, the value of σ parameter affects the mapping of the input data to a higher dimensional space. In this section, an illustrative example is presented to explain the impact of each parameter on the classification performance of SVM.

In this example, the data consists of 500 patterns, and the data was nonlinearly separable as shown in Fig. 1; therefore, the RBF kernel function was used. Table 1 lists the training error rate, number of SVs, and the number of misclassified patterns using different values of C and σ.

3.2.1 The Influence of *C* Parameter

The value of C parameter controls the classification accuracy and the number of SVs. Small C allows the constraints to be easily ignored, while large C makes the constraints difficult to ignore (Wang 2005). In this part from the example, the value of σ parameter was ten. The results of this example with different values of C are given in Table 1.

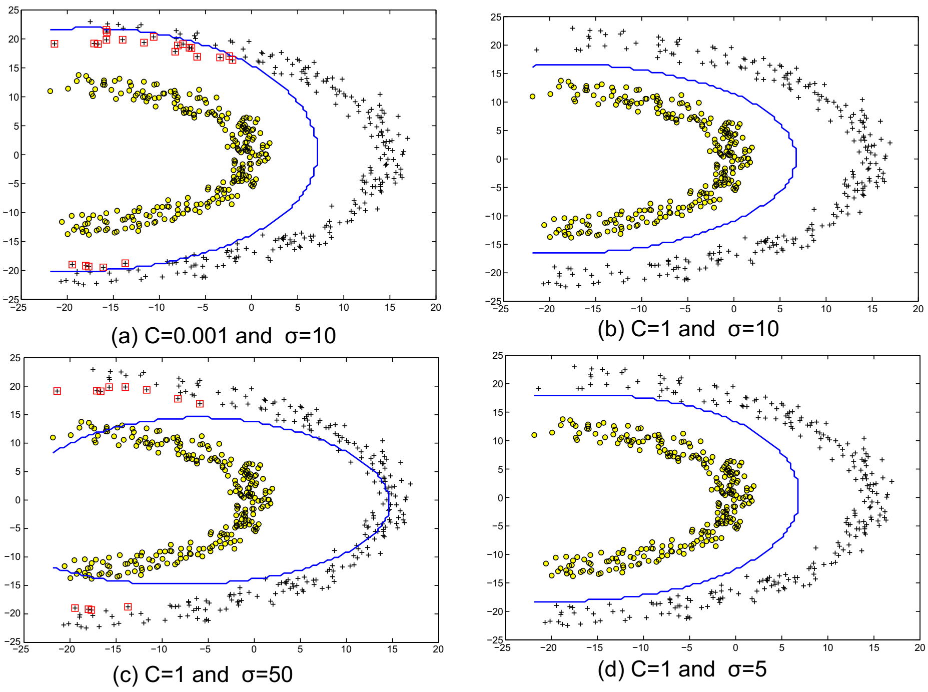


Fig. 1 The effect of the penalty parameter C and kernel parameter σ on the decision boundary of SVM. Decision boundary is in blue line and misclassified patterns are marked with red squares

Table 1 The influence of C and σ on the classification results of SVM

$\sigma = 10$				$C = 1$			
C	Training accuracy (%)	# of Misc. patterns	# SVs	σ	Training accuracy (%)	# of Misc. patterns	# SVs
0.001	50	250	500	0.005	100	0	500
0.01	95	32	434	0.05	100	0	500
0.1	100	0	354	0.5	100	0	365
1	100	0	80	5	100	0	72
10	100	0	34	50	98	12	265
100	100	0	34	500	50	250	479

As shown in Fig. 1a, a small value of C leads to a high number of misclassified patterns and SVs (see, Table 1). In Fig. 1b, the number of misclassified patterns was decreased to zero, and the number of SVs was 80 when $C = 1$. As shown in Table 1, the number of SVs was decreased from 80 to 34 when C was increased from one to ten. Moreover, the number of misclassified patterns reached to its minimum when $C \geq 0.1$.

Generally, when C was small (in our example $C = 0.001$ or $C = 0.01$), the margin was maximized; hence, the number of SVs and misclassified patterns was increased. Thus, very small C may lead to severe underfitting as reported in (Keerthi and Lin 2003). On the contrary, very large C (in our example $C = 100$) minimizes the margin's width and increases the weight of the nonseparable patterns. Thus, one single outlier can affect the boundary, which makes the classifier sensitive to noisy data. Thus, large C may lead to severe overfitting (Keerthi and Lin 2003). In addition, increasing C increases the training time and number of SVs.

From these findings, we can conclude that the value of C parameter is changed through a wide range of values and the optimal value of C can be reached by trying a finite number of values and preserving the value that obtains the minimum classification error (Ben-Hur et al. 2008).

3.2.2 The Influence of σ Parameter

The values of kernel parameter, σ in RBF kernel, affect the partitioning outcome in the feature space. Thus, it has a much greater influence on the classification accuracy than C as reported in (Scholköpf and Smola 2001). The σ parameter defines how far the impact of a single training pattern reaches, with high values meaning close and low values meaning far.

In this example, the value of C parameter was one. The results of this example with different values of σ are given in Table 1. As shown in Table 1, the training error rate was high when the value of σ was 500, while the error rate decreased when $\sigma = 0.005, 0.05, 0.5$, or 5. Moreover, the maximum number of SVs achieved when σ was very low (0.005 or 0.05) or very high (500). In addition, the number of SVs decreased to its minimum when $\sigma = 5$.

From these findings, we can conclude that a very small value of σ tends to make the kernel value for any unknown pattern not very close to a trained pattern. Mathematically, very small σ makes the kernel value, $K(\mathbf{x}_{\text{test}}, \mathbf{x}_i) = \exp(-||\mathbf{x}_{\text{test}} - \mathbf{x}_i||^2/2\sigma^2)$ will be approximately zero, where $\mathbf{x}_{\text{test}}$ is an unknown pattern which is far from a training pattern ($\mathbf{x}_i$). In other words, the kernel function with a very small σ maps the pairwise distance between two far patterns, i.e. $\mathbf{x}_{\text{test}}$ and $\mathbf{x}_i$, to be approximately zero in the new higher dimensional space.

Thus, a very small σ tends to be less bias and more variance this is so-called overfitting; thus, the decision boundary tends to be strict and sharp and the number of SVs increases as shown in Fig. 1d. On the other hand, a very large value of σ maximizes the kernel value, hence increases the number of misclassified patterns and SVs. Therefore, very large σ tends to less variance and more bias this is so-called underfitting; therefore, the decision boundary will be more smooth and flexible (see, Fig. 1c). These findings are in agreement with (Kecman 2001; Wang 2005).

To conclude, from these findings, the σ parameter controlling the flexibility of the resulting SVM in fitting the data. Hence, this parameter may lead to overfitting or underfitting.

3.3 Classical/Standard Particle Swarm Optimization (PSO)

The original PSO was first introduced by Reynolds and simulated by Kennedy and Eberhart (Reynolds 1987; Eberhart and Kennedy 1995). The main goal of PSO is to search in the search space for the positions that are close to the optimal solution(s). In the PSO algorithm, each individual is referred to as a *particle*, and each particle represents one solution to a problem in d-dimensional space; hence, each particle represents one point in the search space. The number of particles (N) is determined by the user.

The particles' positions are initialized randomly, and the position of the ith particle in the tth iteration is denoted by $\mathbf{X}^{(t)}(i) = [\mathbf{X}^{(t)}(i, 1), \ldots, \mathbf{X}^{(t)}(i, d)]$. The movement of each particle depends mainly on two kinds of information. The first kind is the personal experience of the particle. This is represented by the previous best position (the position that achieved the best fitness value by the particle), which is denoted by $\mathbf{P}^{(t)}(i) = [\mathbf{P}^{(t)}(i, 1), \ldots, \mathbf{P}^{(t)}(i, d)]$. The second kind of information is the other particles' experiences (the knowledge of the particles around that particle). This is referred as global best position ($\mathbf{P}^{(t)}(g)$), and it represents the position of the particle that has the best fitness value, where g indicates the index of the best particle among all the particles in the swarm (Santos Coelho 2010).

The particles updates their positions as in Eq. 10.

$$\mathbf{X}^{(t+1)}(i) = \mathbf{X}^{(t)}(i) + \mathbf{V}^{(t)}(i), \tag{10}$$

where $\mathbf{V}^{(t)}(i)$ is the velocity vector of the ith particle in the tth iteration. The velocity is calculated as in Eq. 11 and it consists of the following terms:

1. The current motion or original velocity of that particle ($w\mathbf{V}^{(t)}(i)$), where w is the inertia weight.
2. The position of the previous best position of that particle that is called particle memory or cognitive component. This term is used to adjust the velocity towards the best position visited by that particle ($C_1 r_1(\mathbf{P}^{(t)}(i) - \mathbf{X}^{(t)}(i))$), where C_1 is the cognition learning factor and r_1 is the uniformly generated random number in the range of [0, 1].
3. The position of the best fitness value or social component ($C_2 r_2(\mathbf{P}^{(t)}(g) - \mathbf{X}^{(t)}(i))$) that is used to adjust the velocity towards the global best position, where C_2 indicates the social learning factors and r_2 is the uniformly generated random number in the range of [0, 1] (Hassan et al. 2005)

$$\begin{aligned}\mathbf{V}^{(t+1)}(i) &= w\mathbf{V}^{t}(i) + C_1 r_1(\mathbf{P}^{(t)}(i) - \mathbf{X}^{(t)}(i)) \\ &\quad + C_2 r_2(\mathbf{P}^{(t)}(g) - \mathbf{X}^{(t)}(i)).\end{aligned} \tag{11}$$

As denoted in Eq. 11, the velocity of the particles depends on random variables; thus, the particles are moved randomly. Hence, the motion of the particles is called a random walk (Yang 2014). The PSO algorithm searches for better positions by keep moving the particles towards the global solution (Eberhart and Kennedy 1995; Kennedy 2010).

3.4 Quantum-Behaved Particle Swarm Optimization (QPSO)

In standard PSO, the position and velocity of a particle determine the trajectory, and the particle moves along the determined trajectory in Newtonian mechanics. In quantum mechanics, the trajectory term is meaningless, because the position and velocity of a particle cannot be determined simultaneously according to the uncertainty principle. Thus, in quantum PSO (QPSO), the particles move in a different fashion (Sun and Feng 2004; Sun et al. 2005).

In the QPSO algorithm, the particles are depicted by Schrödinger equation $\psi(x,t)$, instead of position and velocity in the classical or standard PSO algorithm. Hence, the dynamic behavior of the particles in QPSO is widely divergent from PSO, where the exact values of the position and velocity cannot be determined simultaneously (Liu et al. 2005). Employing the Monte-Carlo method, the particles in the QPSO algorithm move according to Eq. 12 (Sun and Feng 2004; Sun et al. 2005; Xi and Sun 2008; Cai et al. 2008).

$$\begin{cases} \mathbf{X}^{(t+1)}(i) = \mathbf{p}(i) + \alpha.|\mathbf{M}_{\text{best}} - \mathbf{X}^{(t)}(i)|.\ln(\frac{1}{u}) & \text{if } k \geq 0.5 \\ \mathbf{X}^{(t+1)}(i) = \mathbf{p}(i) - \alpha.|\mathbf{M}_{\text{best}} - \mathbf{X}^{(t)}(i)|.\ln(\frac{1}{u}) & \text{if } k < 0.5, \end{cases} \tag{12}$$

where α is a design parameter called contraction-expansion (CE) coefficient, u and k are the uniformly generated random numbers in the range of [0, 1], $\mathbf{M}_{\text{best}}$ indicates the global point or mean best of the population and it represents the mean of the previous best positions, i.e., $\mathbf{P}$, of all particles and it is calculated as in Eq. 13, and the $\mathbf{p}$ is the local attractor, which is used to guarantee convergence of the QPSO algorithm, the value of p is defined as in Eq. 14 (Santos Coelho 2010).

$$\begin{aligned} \mathbf{M}_{\text{best}}^{(t)} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{P}^{(t)}(i) \\ &= \left(\frac{1}{N}\sum_{i=1}^{N}P^{(t)}(i,1), \frac{1}{N}\sum_{i=1}^{N}P^{(t)}(i,2), \ldots, \frac{1}{N}\sum_{i=1}^{N}P^{(t)}(i,d)\right), \end{aligned} \tag{13}$$

$$\mathbf{p}(\mathbf{i}) = \phi\mathbf{P}^{(t)}(i) + (1-\phi)\mathbf{P}^{(t)}(g), \tag{14}$$

where

$$\phi = \frac{C_1 r_1}{C_1 r_1 + C_2 r_2}. \tag{15}$$

The CE parameter (α) controls the speed of convergence. Some studies used fixed α and the results proved that this parameter is sensitive to the population size and the maximum number of iterations. Hence, fixed α is not practically efficient. Sun et al. proposed to use time-varying CE parameter instead of using a fixed value, and they found that decreasing the value from α_1 to α_0 ($\alpha_0 < \alpha_1$) in the course of iterations is simple and efficient than fixed value (Sun et al. 2012). In this research, the value of α is computed as follows:

$$\alpha = \alpha_0 + (\text{Max}_{\text{iter}} - t).(\alpha_1 - \alpha_0)/\text{Max}_{\text{iter}} \tag{16}$$

where α_0 is the initial value of α, α_1 is the final value of α, t is the current iteration, and Max_{iter} is the maximum number of iterations. The value of α_0 was suggested to be smaller than 0.6, and the value of α_1 was suggested to be larger than 0.8 and smaller than 1.2 (Sun et al. 2012).

The steps of QPSO algorithm are listed in Algorithm 1.

Algorithm 1 Quantum particle swarm optimization (QPSO)

```
1:  Initialize the particles' positions ($\mathbf{X}^{(t)}(i)$) and the number of particles $N$.
2:  Set $\mathbf{P}^{(t)}(i) = \mathbf{X}^{(t)}(i)$ , $\forall\, i = 1, 2, \ldots, N$.
3:  while ($t < Max_{iter}$) do
4:      for all $i = 1$ to $N$ do
5:          Calculate $\mathbf{M}_{best}$ as in Eq. 13.
6:          Evaluate the fitness function for the current position ($\mathcal{F}(X^{(t)}(i))$).
7:          if ($\mathcal{F}(\mathbf{X}^{(t)}(i)) < \mathcal{F}(\mathbf{P}^{(t)}(i))$) then
8:              $\mathbf{P}^{(t)}(i) = \mathbf{X}^{(t)}(i)$.
9:          end if
10:         $\mathbf{P}^{(t)}(g) = min(\mathcal{F}(\mathbf{P}^{(t)}(i)))$.
11:         for $j = 1$ to $d$ do
12:             $\phi$ = rand(0,1).
13:             $p(i, j) = \phi P^{(t)}(i, j) + (1 - \phi) P^{(t)}(g, j)$.
14:             $u$ = rand(0,1).
15:             $k$ = rand(0,1).
16:             if ($k \geq 0.5$) then
17:                 $X^{(t)}(i, j) = p(i, j) - \beta.|(M_{best}(i, j) - X^{(t)}(i, j)|.ln(1/u)$.
18:             else
19:                 $X^{(t)}(i, j) = p(i, j) + \beta.|(M_{best}(i, j) - X^{(t)}(i, j)|.ln(1/u)$.
20:             end if
21:         end for
22:     end for
23:     Stop the algorithm if a sufficiently good fitness function is met.
24: end while
```

3.4.1 QPSO vs. PSO

There are some characteristics that make QPSO outperforms PSO such as follows:

- In QPSO, the exponential distribution of the particles' positions makes QPSO globally convergent.
- The PSO particles converge to the global best solution. On the other hand, in QPSO, the particles cannot converge to the global best position without considering their colleagues. Equation 12 indicates that the distance between the current position and $\mathbf{M}_{\text{best}}$ position determines the position for the particle's in the next iteration. Hence, in QPSO, the $\mathbf{M}_{\text{best}}$ may be pulled away from the global best position ($\mathbf{P}^{(t)}(g)$) by the particles which are far from the global best position, called lagged particles. When the lagged particles converging to $\mathbf{P}^{(t)}(g)$, the $\mathbf{M}_{\text{best}}$ will be approaching $\mathbf{P}^{(t)}(g)$ slowly. The distance between M_{best} and the previous best positions of particles near $\mathbf{P}^{(t)}(g)$ is decreased slowly, which decelerate the convergence of particles near $\mathbf{P}^{(t)}(g)$; hence, the particles can wait and explore globally around $\mathbf{P}^{(t)}(g)$ until the lagged particles are

close to $\mathbf{P}^{(t)}(g)$, this called waiting phenomena. Therefore, QPSO never abandons any lagged particle and hence QPSO seems to be more cooperative and intelligent. Simply, $\mathbf{M}_{\text{best}}$ enhances the search capability of the particles and makes them search globally for the optimal solution (Sun et al. 2012).

Figure 2 shows the difference between PSO and QPSO. As shown, the black circle represents the global best position, blue circles represent the lagged particles, and red circles represent the other particles. The dashed arrows around the circles, in QPSO, represent the possible direction of the particles; the solid arrow with a big arrowhead points to the direction in which the particle moves with high probability. As shown in the left panel in Fig. 2, the particles in PSO algorithm converges to the global best position without waiting for their colleagues. Hence, the influence of the lagged particle on the other particles is very little, and if the lagged particles fail to find positions better than $\mathbf{P}^{(t)}(g)$ during iteration, their influence on the other particles will be null. On the contrary, in the QPSO algorithm (as shown in the right panel in Fig. 2), the lagged particles have a great impact on the other particles through using $\mathbf{M}_{\text{best}}$; hence, the particles around $\mathbf{P}^{(t)}(g)$ can fly in any direction. However, the particles move in the direction toward $\mathbf{P}^{(t)}(g)$ with higher probability since the overall tendency of the particles' movements is convergence to $\mathbf{P}^{(t)}(g)$. In the next section, an illustrative example will be presented to explain the difference between PSO and QPSO algorithms.

3.4.2 QPSO vs. PSO: an Illustrative Example

In this section, a simple example is introduced to show the difference between the QPSO and PSO algorithms. In this example, the PSO and QPSO algorithms were used to search for the optimal solution and the fitness function was as follows:

$$\min:\ \mathcal{F}(x)=\sum_{i=1}^{d}x_i^2\ ,\ -5\leq x_i\leq 5 \tag{17}$$

In this example, for both algorithms, the number of particles was 20, and the maximum number of iterations was 30.

Figure 3 shows the average distance (AD) between the lagged particles and the global best position in the PSO and QPSO algorithms. In this example, the farthest five particles

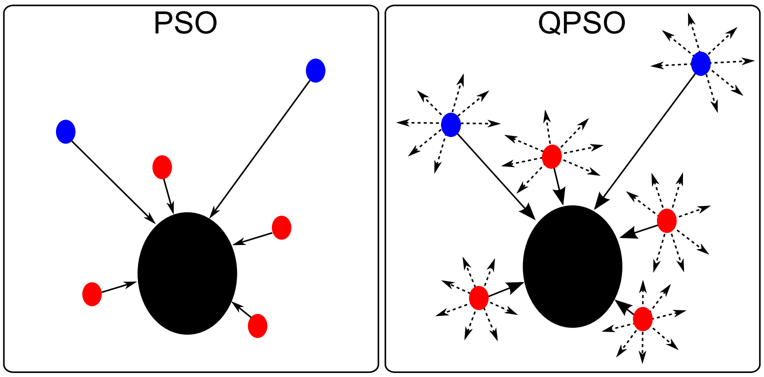


Fig. 2 The movements of particles in PSO and QPSO algorithms

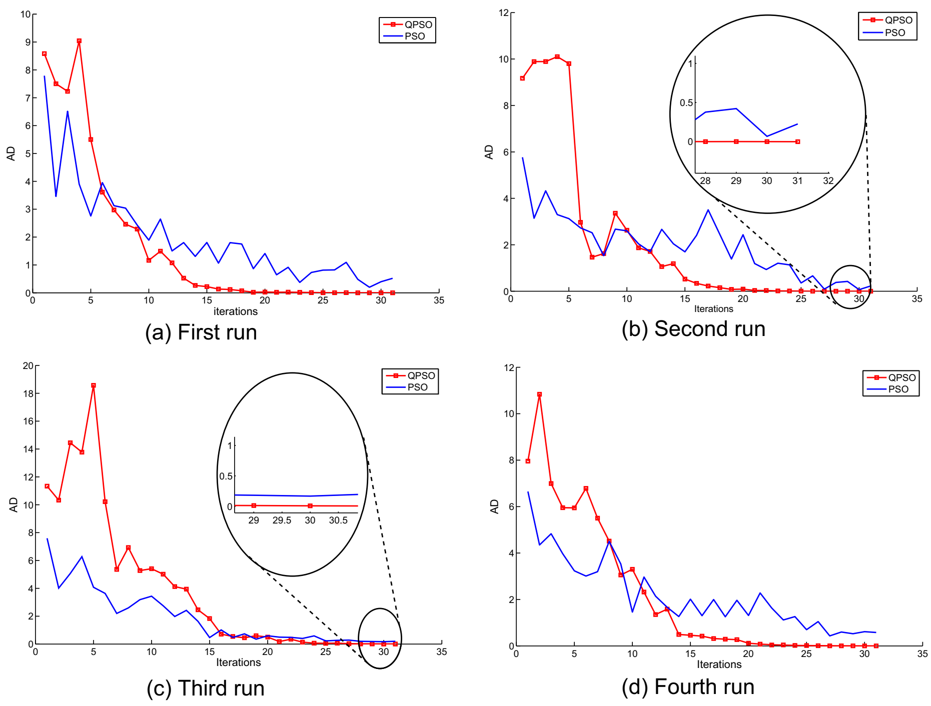


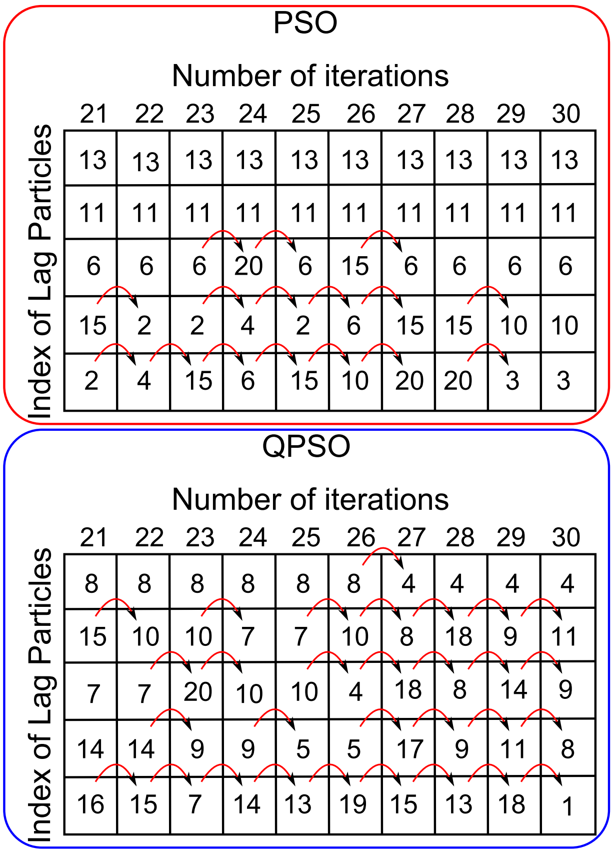
Fig. 3 The average distance (AD) between the lagged particles and the global best position in the PSO and QPSO algorithms

from the global best position represent the lag particles. As shown, in different four runs, as the iterations proceed, the lagged particles of QPSO algorithm become closer to the $\mathbf{P}^{(t)}(g)$ than PSO.

From this finding, it can be concluded that the particles in the QPSO algorithm were more cooperative as they help the lagged particles to be closer to the global best position through using $\mathbf{M}_{\text{best}}$ parameter; hence, the distance between the lagged particles and $\mathbf{P}^{(t)}(g)$ decreased as the iterations proceed.

Figure 4 shows the indices of the lagged particles during the last ten iterations, where the top index represents the farthest particle from $\mathbf{P}^{(t)}(g)$, and the red arrow indicates the change of index. As shown, in the PSO panel, as the iterations proceed, the top two indices were stable and they did not change. Moreover, as the iterations proceed, there was a small number of indices changed, and there were only nine different indices during the last ten iterations. On the other hand, in the QPSO algorithm, the indices were changed more than in PSO. As shown, in the QPSO panel, the first index was 8, and then it changed to 4, and the second index was changed many times during the iterations, and there were 17 different indices during the last 10 iterations.

From this finding, it can be concluded that due to the noncooperative behavior of the PSO algorithm, the lagged particles may remain constant, i.e., far from $\mathbf{P}^{(t)}(g)$. This noncooperative behavior has a significant negative impact on the performance of the PSO algorithm. While, in the QPSO algorithm, $\mathbf{M}_{\text{best}}$ parameter helps the lagged particles to converge to $\mathbf{P}^{(t)}(g)$; hence, QPSO outperformed PSO as reported in (Sun et al. 2011, 2012).



PSO

Number of iterations

Index of Lag Particles

21	22	23	24	25	26	27	28	29	30
13	13	13	13	13	13	13	13	13	13
11	11	11	11	11	11	11	11	11	11
6	6	6	20	6	15	6	6	6	6
15	2	2	4	2	6	15	15	10	10
2	4	15	6	15	10	20	20	3	3

QPSO

Number of iterations

Index of Lag Particles

21	22	23	24	25	26	27	28	29	30
8	8	8	8	8	8	4	4	4	4
15	10	10	7	7	10	8	18	9	11
7	7	20	10	10	4	18	8	14	9
14	14	9	9	5	5	17	9	11	8
16	15	7	14	13	19	15	13	18	1

Fig. 4 The indices of lag particles in the PSO and QPSO algorithms during the last ten iterations

4 The Proposed Model: QPSO-SVM

In this section, we describe the proposed QPSO-SVM model (as shown in Fig. 5) to find optimal values of SVM parameters. The detailed explanation is as follows:

4.1 Data Preprocessing

The first step in our model is the data preprocessing. This step adopts linear scaling to (1) avoid features in greater numeric ranges dominating those in smaller ranges and (2) avoid numerical difficulties during the calculation (Zhao et al. 2011). Each feature can be scaled to the range [0, 1] as follows:

$$v' = \frac{v - \min}{\max - \min}, \tag{18}$$

where v is the original value, min and max are the lower and upper bounds of the feature value, respectively, and v' is the scaled value.

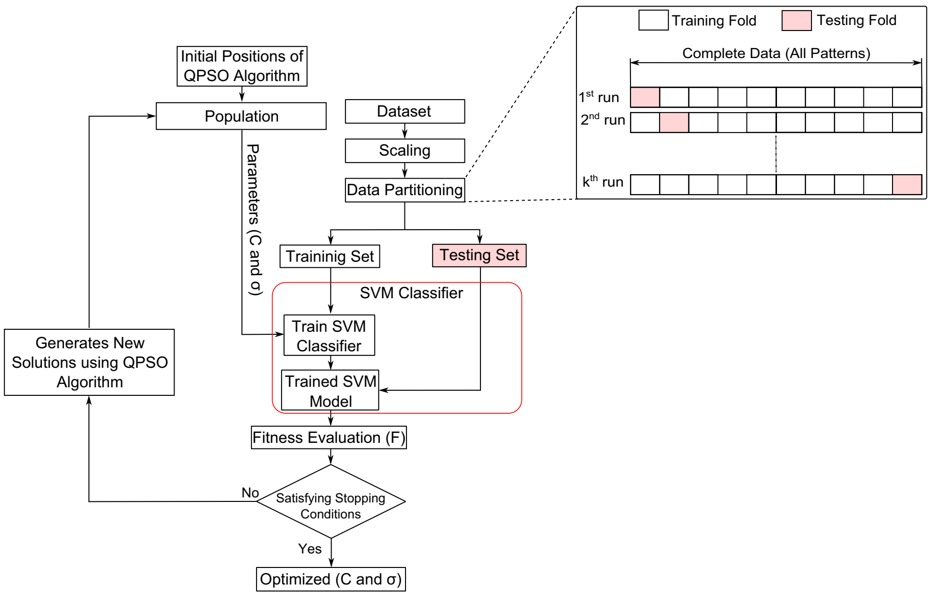


Fig. 5 Flowchart of the proposed model (QPSO-SVM)

4.2 Data Partitioning

In this model, the data was partitioned using the k-fold cross-validation method. In k-fold cross-validation, the original data was randomly partitioned into k subsets of (approximately) equal size, and the experiment is run k times as shown in Fig. 5. For each run, one subset was selected as the testing set, and the other $k-1$ subsets were utilized as the training set. The average of the k results from the folds can then be calculated to produce a single estimation.

4.3 Parameters' Initialization

In this step, the parameters of the QPSO algorithm including the number of particles, maximum number of iterations, and initial positions were initialized. In the proposed model, the QPSO algorithm provides the SVM classifier with the values of C and σ to train SVM using the training set. Thus, there were only two parameters in our problem; thus, the search space is two-dimensional, and each position represents a combination between the C and kernel parameter. The particles' positions were initialized randomly. The searching range of parameter C of SVM was bounded by $C_{\min} = 0.01$ and $C_{\max} = 3000$, and the searching range of σ parameter was bounded by $\sigma_{\min} = 0.01$ and $\sigma_{\max} = 100$ (Lin et al. 2008). Increasing these limits enlarges the search space; thus, more particles are needed to search for the optimal solution, which results in more computation and slow convergence rate.

4.4 Fitness Evaluation

For each position, the training set is used to train the SVM classifier, while the testing set is used to calculate misclassification or error rate, which is defined as the ratio between the number of misclassified patterns (N_e) to the total number of testing patterns (N) as denoted in Eq. 19. The optimal solution ($X*$) is located in a position ($X_i \in \mathcal{R}^2$) where the values of C and σ at that location achieve the minimum testing error rate.

$$\text{Minimize} : \mathcal{F} = \frac{N_e}{N} \tag{19}$$

4.5 Termination Criteria

The positions of particles were updated according to Eq. 12. In the proposed model, the QPSO algorithm is terminated when a maximum number of iterations are reached or when the best solution is not updated for a given number of iterations. In our experiments, the maximum number of repetitions of the best solution was ten.

5 Experimental Results and Discussion

The experiments were carried out to evaluate the performance of the proposed QPSO-SVM algorithm for the parameter optimization of SVM.

5.1 Experimental Setup

To get an unbiased comparison of CPU times, all experiments are performed using the same PC with the detailed settings as shown in Table 2.

To evaluate the proposed algorithm, seven standard classification datasets were used. The datasets were obtained from University of California at Irvin (UCI) Machine Learning Repository (Blake and Merz 1998). The descriptions of all datasets are listed in Table 3. These datasets are widely utilized to compare the performance of different classification problems in the literature. As shown in Table 3, the iris, wine, and glass datasets have more than two classes; thus, one-against-all (1AA) multiclass SVM method was used (Friedrichs and Igel 2005). Since the number of patterns in each class is not a multiple of ten as shown in Table 3, the dataset cannot be partitioned fairly using k-fold cross-validation. However, the ratio between the number of patterns in the training and testing sets was maintained as closely as possible to 9:1. We must point out that the abbreviations $D_1, D_2, \ldots, D_7$ in Table 3 are short for the datasets listed in the Table.

Table 2 The detailed settings

Name	Detailed settings
Hardware	
CPU	Core (TM) i5-2400
Frequency	3.10 GHz
RAM	4G
Hard Drive	160 GB
Software	
Operating system	Window 7
Language	MATLAB R2012a (7.14)

Table 3 Datasets description

Dataset	Dimension	# Patterns	# Classes
Wine (D1)	13	178	3
Ionosphere (D2)	34	351	2
Iris (D3)	4	150	3
Liver-disorders (D4)	6	345	2
Sonar (D5)	60	208	2
Glass (D6)	9	214	7
Pima Indians diabetes (D7)	8	768	2

5.2 Parameter Setting for QPSO

Tuning the parameters for any optimization algorithm is important as designing the algorithm itself. In this section, the effect of the number of particles and the maximum number of iterations on the testing error rate and computational time of the proposed model (QPSO-SVM) were investigated using Iono dataset.

5.2.1 Number of Particles

In this section, the effect of the number of particles on the testing error and computational time of the proposed model was tested when the number of particles ranged from 5 to 30. The results of this experiment are shown in Fig. 6. From the figure, it is observed that increasing the number of particles reduces the testing error, but more computational time was needed. Moreover, the minimum error rate reached to its minimum when the number of particles was 20.

5.2.2 Number of Iterations

The number of iterations/generations also have a great impact on the performance of the proposed model. In this section, the effect of the number of iterations on the testing error and computational time of the proposed model was tested when the number of iterations was ranged from 10 to 100. The results of this experiment are shown in Fig. 7. As shown in

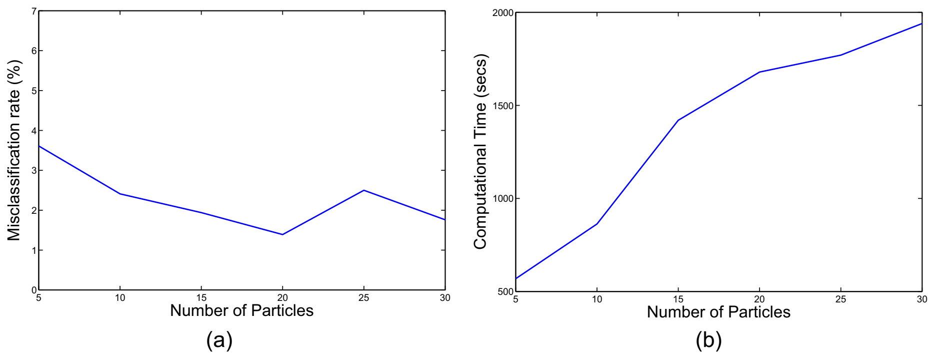


Fig. 6 Effect of the number of particles on the performance of the QPSO-SVM model using Iono dataset: **a** testing error rate of the QPSO-SVM model with different numbers of particles; **b** computational time of the QPSO-SVM model using different numbers of particles

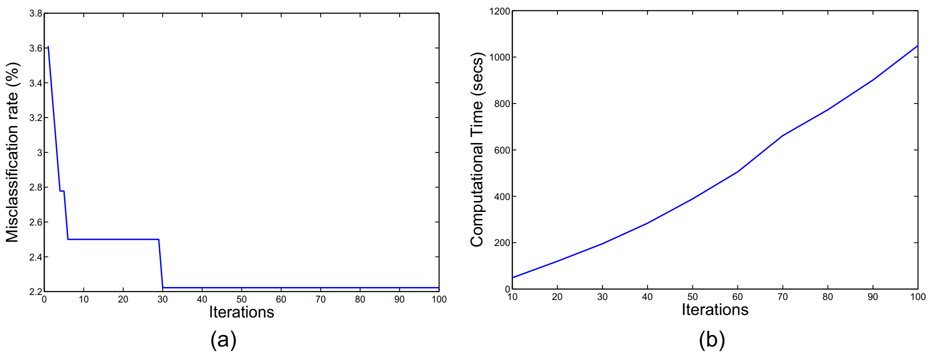


Fig. 7 Effect of the number of iterations on the performance of the QPSO-SVM model using Iono dataset. **a** Testing error rate of the QPSO-SVM model with different numbers of iterations; **b** computational time of the QPSO-SVM model using different numbers of iterations

the figure, it can be noticed that, when the number iterations was increased, the testing error was decreased until it reached an extent at which increasing the number of iterations did not affect the testing error. In addition, the computational time increased when the number of iterations was increased.

According to the other parameters of QPSO, Jun Sun et al. suggested that QPSO algorithm can obtain a global convergence when $\alpha_0 < 0.6$ and $1.2 > \alpha_1 > 0.8$ (Sun et al. 2012). On the basis of the above parameter analysis and research results, Table 4 lists the detailed settings for the QPSO algorithm that were used in the proposed model.

5.3 Experimental Results

In this section, three experiments were conducted. The aim of the first experiment is to compare the proposed model, i.e., QPSO-SVM with PSO-SVM. The goal of the second experiment is to compare the QPSO-SVM with the grid search SVM. The last and third experiment was conducted to compare QPSO-SVM with one of the well-known optimization algorithms: GA. To compare the classification error rate of the proposed algorithm with the other algorithms, we used the nonparametric Wilcoxon signed-rank test for all datasets.

5.3.1 QPSO vs. Classical PSO

This experiment was conducted to compare our proposed QPSO-SVM algorithm with the PSO-SVM that was proposed in (Lin et al. 2008). In this experiment, the C and σ parameters were optimized by using our proposed algorithm. Table 5 shows the average classification

Table 4 The initial parameters of QPSO algorithm

Parameter	Value
α_0	0.5
α_1	0.9
Population size	20
Maximum number of iterations	50
Problem dimension	2

Table 5 Comparison between the proposed QPSO-SVM algorithm and the PSO-SVM algorithm that was proposed by (Lin et al. 2008) (%)

Dataset	QPSO-SVM	PSO-SVM	p value for Wilcoxon testing
D_1	7.85 ± 5.45	8.89 ± 6.08	< 0.005
D_2	2.22 ± 3.15	2.5 ± 2.76	< 0.005
D_3	1.33 ± 4.22	2.0 ± 4.50	< 0.005
D_4	21.70 ± 3.07	23.29 ± 5.47	< 0.005
D_5	7.62 ± 7.17	8.62 ± 7.0	0.1285
D_6	18.70 ± 8.42	20.23 ± 3.46	< 0.005
D_7	21.33 ± 5.45	21.33 ± 6.08	< 0.005

error rate obtained by the QPSO and PSO algorithms in this experiment, and the obtained results are illustrated with the form of *average* $\pm$ *standard deviation*. As shown, the QPSO-SVM algorithm yielded lower classification error rates than the PSO-SVM algorithm in all datasets. Moreover, the p value for D5 (Sonar dataset) is larger than the predicted statistical significance level of 0.005, but the other p values are smaller than the significance level of 0.005. Furthermore, the optimization processes of QPSO-SVM and PSO-SVM are given in Fig. 8. The values shown in these figures are the average of all cross-validation results, and both algorithms started from the same initial solutions. As shown in the figure, the QPSO-SVM algorithm achieved results better than the PSO-SVM algorithm. Figure 8c, e shows clearly how the PSO was trapped in one of the local minima solutions, while the QPSO algorithm escaped and converged to better solutions. Moreover, the figure shows how the QPSO converged faster than the PSO algorithm. For example, as in Fig. 8g, the QPSO and PSO algorithms achieved the same solution, but the QPSO and PSO algorithms reached to this solution after 24 and 48 iterations, respectively. This reflects the convergence speed of the QPSO algorithm compared to PSO.

Generally, from Table 5 and Fig. 8, compared with the PSO-SVM, the proposed QPSO-SVM algorithm achieved lower classification error rate. This is due to (1) the exponential distribution of the particles' positions and (2) the cooperative behavior of the QPSO algorithm as mentioned in Sections 3.4.1 and 3.4.2. Therefore, it is no surprise that QPSO is efficient than classical PSO.

5.3.2 QPSO-SVM vs. Grid Search

This experiment was performed to compare the proposed QPSO-SVM algorithm with the grid search SVM algorithm. Table 6 shows the test error rate and computational time of this experiment. k-fold cross-validation ($k = 10$) was used to estimate the test error rate of each approach, and the obtained results are illustrated with the form of *average* $\pm$ *standard deviation*. As shown in the table, the cost time of the proposed algorithm was much lower than the grid search algorithm. As shown in Table 6, the p value for D_5 (Sonar dataset) is larger than the predicted statistical significance level of 0.005, but the other p values are smaller than the significance level of 0.005. Generally, compared with the grid search algorithm, the proposed QPSO-SVM algorithm achieved lower classification error rates.

5.3.3 QPSO-SVM vs. Other Optimization Algorithms

This experiment was conducted to compare our proposed QPSO-SVM algorithm with one of the well-known optimization algorithms that was proposed in (Huang and Wang 2006) to

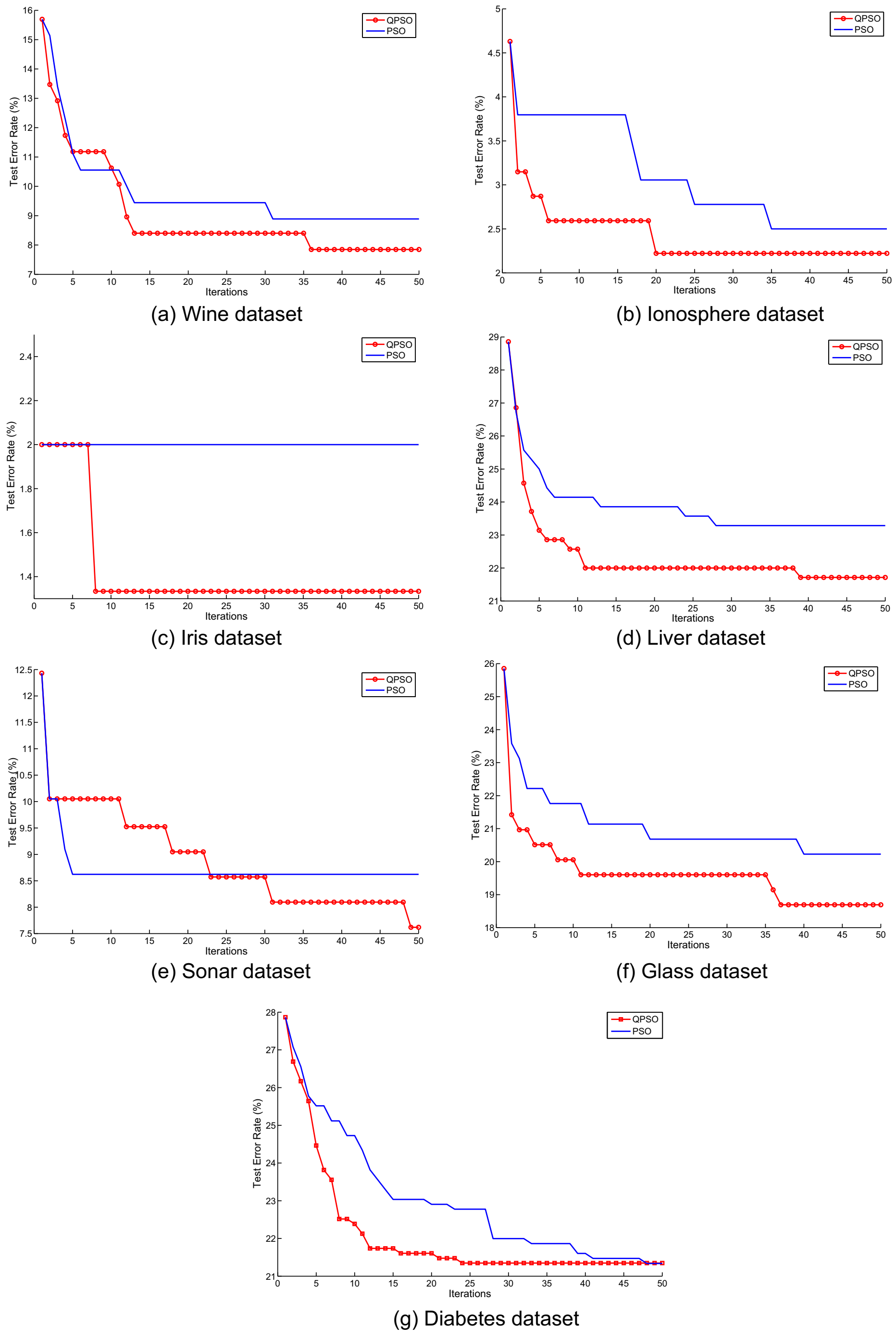


Fig. 8 Performance comparison using the datasets listed in Table 3

Table 6 Results of the proposed QPSO-SVM algorithm and the grid search SVM algorithm

Dataset	Grid search SVM		QPSO-SVM		p value for Wilcoxon testing
	Cost time (s)	Test error (%)	Cost time (s)	Test error (%)	
D_1	398.23	9.12 ± 4.89	287.35	7.85 ± 5.45	< 0.005
D_2	925.3	2.4 ± 3.45	822.32	2.22 ± 3.15	< 0.005
D_3	237.6	2.54 ± 4.42	165.5	1.33 ± 4.22	< 0.005
D_4	890.3	22.67 ± 4.30	765.2	21.70 ± 3.07	< 0.005
D_5	597.7	7.83 ± 7.40	489.6	7.62 ± 7.17	0.0054
D_6	567.4	20 ± 10.23	408.3	18.70 ± 8.42	< 0.005
D_7	1975.3	23.14 ± 6.25	1608.5	21.33 ± 5.45	< 0.005

optimize the values of C and σ parameters. Table 7 shows the results of this experiment. As shown, QPSO-SVM algorithm achieved lower classification error rates than the GA-SVM algorithm.

5.3.4 QPSO-SVM vs. Conventional Classifiers

The aim of this experiment is to test the proposed QPSO-SVM model using different classifiers. In this experiment, the QPSO-SVM algorithm was compared with k-nearest neighbor (k-NN) (Dudani 1976), multi-layer perceptron (MLP) (Pal and Mitra 1992), and linear discriminant analysis (LDA) (Tharwat 2016a) classifiers. The value of k in k-NN was three and in MLP, a hidden layer with 15 nodes and 1000 epochs were used. The results of this experiment are summarized in Table 8.

As shown in Table 8, the proposed QPSO-SVM algorithm obtained the best results compared to the other classifiers, while the kNN classifier achieved the worst results. This is because (1) in k-NN, there is no training phase and the testing sample assigned to the class which has the maximum neighbors, (2) NN has different parameters and these parameters also need to be tuned, and (3) LDA has the small sample problem (SSS); and this problem can be solved by reducing the dimensions using one of the dimensionality reduction methods such as the principal component analysis (PCA) (Tharwat 2016b) algorithm. Hence, PCA removes some information which may have discriminative data (Tharwat et al. 2017).

To conclude, the developed QPSO-SVM algorithm yielded more appropriate parameters, giving lower classification error rate across different datasets. Even so, we still

Table 7 Comparison between the QPSO-SVM algorithm and the GA-SVM algorithm that was proposed by (Huang and Wang 2006) (%)

Dataset	QPSO-SVM	GA-SVM	p value for Wilcoxon testing
D_1	7.85 ± 5.45	9.69 ± 6.44	< 0.005
D_2	2.22 ± 3.15	2.5 ± 3.50	< 0.005
D_3	1.33 ± 4.22	2.70 ± 3.50	< 0.005
D_4	21.70 ± 3.07	22.95 ± 5.23	< 0.005
D_5	7.62 ± 7.17	9.72 ± 7.34	< 0.005
D_6	18.70 ± 8.42	22.24 ± 4.48	< 0.005
D_7	21.33 ± 5.45	23.56 ± 6.51	< 0.005

Table 8 Comparison between the QPSO-SVM algorithm and NN, LDA, and k-NN classifiers (%)

Datasets	QPSO-SVM	NN	LDA	k-NN
D1	7.85 + 5.45	10.75 + 4.46	11.26 + 3.45	13.45 + 6.23
D2	2.22 + 3.15	4.32 + 3.68	3.93 + 2.76	6.32 + 3.12
D3	1.33 + 4.22	1.33 + 6.43	3.45 + 2.21	4.15 + 2.36
D4	21.70 + 3.07	23.64 + 4.32	23.51 + 5.23	26.18 + 4.62
D5	7.6 + 7.7	8.52 + 6.98	9.54 + 6.23	11.23 + 4.75
D6	18.70 + 8.42	19.23 + 7.98	20.45 + 8.23	20.15 + 8.05
D7	21.33 + 5.45	23.26 + 6.45	25.23 + 7.64	24.35 + 7.49

cannot guarantee that QPSO-SVM must perform well and outperform other methods in several applications. In fact, many factors may have a great impact on the quality of the proposed algorithm, such as the number of classes, the number of training samples, representativeness, the number of features, and diversity of training sets.

6 Conclusion and Future Work

The SVM classifier is widely used in different applications. However, the parameters of SVM affect the test error rate. This study proposes a quantum-behaved based approach (QPSO-SVM) that can search for the optimal values of SVM parameters that minimize the classification error rate. In this research, different experiments were conducted to compare the proposed QPSO-SVM algorithm with the grid search SVM, PSO-SVM, and GA-SVM algorithms by applying many standard classification datasets. The results of this study showed that the QPSO-SVM algorithm achieved competitive results.

Several directions for future studies are suggested. First, our fitness function is not suitable for imbalanced datasets, where the number of patterns of one class (majority class) outnumber the number of the other class(es) (minority class(es)). This is because the accuracy rate does not distinguish between the number of correct labels of different classes. Thus, in the imbalanced datasets, the accuracy may lead to erroneous conclusions (He and Garcia 2009). Therefore, different assessment methods such as geometric mean (GM) can be used in the fitness function. Second, due to the performance of QPSO, it would be worthwhile to explore the potential of QPSO to other classifiers. This is currently being investigated by the authors of this paper.

References

Ali, S., & Smith, K. (2003). Automatic parameter selection for polynomial kernel. In *Proceedings of IEEE International Conference on Information Reuse and Integration, (IRI 2003), Lens, France, October*, (Vol. 27-29 pp. 243-249).

Bashir, Z., & El-Hawary, M. (2009). Applying wavelets to short-term load forecasting using PSO-based neural networks. *IEEE Transactions on Power Systems*, *24*(1), 20–27.

Ben-Hur, A., Ong, C.S., Sonnenburg, S., Schölkopf, B., Rätsch, G. (2008). Support vector machines and kernels for computational biology. *PLoS Comput Biol*, *4*(10), e1000173.

Blake, C., & Merz, C.J. (1998). {UCI} repository of machine learning databases repository of machine learning databases.

Byvatov, E., & Schneider, G. (2002). Support vector machine applications in bioinformatics. *Applied Bioinformatics*, *2*(2), 67–77.

Cai, Y., Sun, J., Wang, J., Ding, Y., Tian, N., Liao, X., et al. (2008). Optimizing the codon usage of synthetic gene with QPSO algorithm. *Journal of Theoretical Biology*, *254*(1), 123–127.

Chander, A., Chatterjee, A., Siarry, P. (2011). A new social and momentum component adaptive PSO algorithm for image segmentation. *Expert Systems with Applications*, *38*(5), 4998–5004.

Chapelle, O., Vapnik, V., Bousquet, O. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*(1-3), 131–159.

Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73.

Dudani, S.A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(4), 325–327.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS'95. pp. 39–43.*

Friedrichs, F., & Igel, C. (2005). Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, *64*, 107–117.

Hassan, R., Cohanim, B., De, W.eck., Venter, O. (2005). G A comparison of particle swarm optimization and the genetic algorithm. In *Proceedings of the* 1^{st} *AIAA multidisciplinary design optimization specialist conference, Honolulu, Hawaii, April 23-26* (pp. 1–13).

He, H., & Garcia, E.A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.

Huang, C.L., & Wang, C.J. (2006). A GA-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with Applications*, *31*(2), 231–240.

Kecman, V. (2001). *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. Cambridge: MIT Press.

Keerthi, S.S., & Lin, C.J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, *15*(7), 1667–1689.

Kennedy, J. (2004). Probability and dynamics in the particle swarm. In *Congress on Evolutionary Computation.*

Kennedy, J. (2005). Dynamic-probabilistic particle swarms. In *Proceedings of the 7th Annual Conference On Genetic And Evolutionary Computation, pp. 201–207.*

Kennedy, J. (2010). Particle swarm optimization. In *Encyclopedia of Machine Learning. Springer, pp. 760–766.*

Krohling, R.A. (2004). Gaussian swarm: a novel particle swarm optimization algorithm. In *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, (Vol. 1 pp. 372-376).

Krohling, R.A., & Santos Coelho, L.dos. (2006). PSO-E: particle swarm with exponential distribution. In *2006 IEEE International Conference on Evolutionary Computation, pp 1428–1433.*

Li, S., Wang, R., Hu, W., Sun, J. (2007). A new QPSO based BP neural network for face detection. In *Fuzzy Information and Engineerin. Springer, pp. 355–363.*

Liang, J.J., Qin, A.K., Suganthan, P.N. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, *10*(3), 281–295.

Lin, S.W., Ying, K.C., Chen, S.C., Lee, Z.J. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Systems with Applications*, *35*(4), 1817–1824.

Liu, B., Wang, L., Jin, Y.H., Tang, F. (2005). Huang D X. *Chaos, Solitons & Fractals*, *25*(5), 1261–1271.

Liu, J., Xu, W., Sun, J. (2005). Quantum-behaved particle swarm optimization with mutation operator. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05).*

Maitra, M., & Chatterjee, A. (2008). A hybrid cooperative–comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding. *Expert Systems with Applications*, *34*(2), 1341–1350.

Merwe, D., Van der, Engelbrecht, A.P. (2003). Data clustering using particle swarm optimization. In *The 2003 Congress on Evolutionary Computation(CEC'03)*, (Vol. 1 pp. 215-220).

Mikki, S.M., & Kishk, A.A. (2006). Quantum particle swarm optimization for electromagnetics. *IEEE Transactions on Antennas and Propagation*, *54*(10), 2764–2775.

Miyatake, M., Veerachary, M., Toriumi, F., Fujii, N., Ko, H. (2011). Maximum power point tracking of multiple photovoltaic arrays: a PSO approach. *IEEE Transactions on Aerospace and Electronic Systems*, *47*(1), 367–380.

Omkar, S., Khandelwal, R., Ananth, T., Naik, G.N. (2009). Quantum behaved Particle Swarm Optimization (QPSO) for multi-objective design optimization of composite structures. *Expert Systems with Applications*, *36*(8), 11312–11322.

Pal, S.K., & Mitra, S. (1992). Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks*, *3*(5), 683–697.

Panda, S., & Padhy, N.P. (2008). Optimal location and controller design of STATCOM for power system stability improvement using PSO. *Journal of the Franklin Institute*, *345*(2), 166–181.

Reynolds, C.W. (1987). Flocks, herds and schools: a distributed behavioral model. *ACM Siggraph Computer Graphics*, *21*(4), 25–34.

Richer, T.J., & Blackwell, T.M. (2006). The Lévy particle swarmvy particle swarm. In *2006 IEEE International Conference on Evolutionary Computation, pp. 808–815*.

Santos Coelho, L.d.os. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications*, *37*(2), 1676–1683.

Santos Coelho, L.d.os., & Krohling, R.A. (2005). Predictive controller tuning using modified particle swarm optimization based on Cauchy and Gaussian distributions. In *Soft Computing: Methodologies and Applications. Springer, pp. 287–298*.

Scholköpf, B., & Smola, A.J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Cambridge: MIT Press.

Subasi, A. (2013). Classification of EMG signals using PSO optimized SVM for diagnosis of neuromuscular disorders. *Computers in Biology and Medicine*, *43*(5), 576–586.

Sun, J., Fang, W., Palade, V., Wu, X. (2011). Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point. *Applied Mathematics and Computation*, *218*(7), 3763–3775.

Sun, J., Fang, W., Wu, X., Palade, V. (2012). Quantum-behaved particle swarm optimization: analysis of individual particle behavior and parameter selection. *Evolutionary Computation*, *20*(3), 349–393.

Sun, J., & Feng, B. (2004). Particle swarm optimization with particles having quantum behavior. In *Congress on Evolutionary Computation*.

Sun, J., Xu, W., Feng, B. (2004). A global search strategy of quantum-behaved particle swarm optimization. In *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, (Vol. 1 pp. 111-116).

Sun, J., Xu, W., Feng, B. (2005). Adaptive parameter control for quantum-behaved particle swarm optimization on individual level. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, (Vol. 4 pp. 3049-3054).

Tharwat, A. (2016). Linear vs. quadratic discriminant analysis classifier: a tutorial. *International Journal of Applied Pattern Recognition*, *3*(2), 145–180.

Tharwat, A. (2016). Principal component analysis-a tutorial. *International Journal of Applied Pattern Recognition*, *3*(3), 197–240.

Tharwat, A., Gabel, T., Hassanien, A.E. (2017). Parameter optimization of support vector machine using dragonfly algorithm. In *International Conference on Advanced Intelligent Systems and Informatics, pp. 309–319*.

Tharwat, A., Gaber, T., Ibrahim, A. (2017). Linear discriminant analysis: a detailed tutorial. *AI Communications*, *30*(2), 169–190.

Tharwat, A., & Hassanien, A.E. (2018). Chaotic antlion algorithm for parameter optimization of support vector machine. *Applied Intelligence*, *48*(3), 670–686.

Tharwat, A., Hassanien, A.E., Elnaghi, B.E. (2016). A BA-based algorithm for parameter optimization of support vector machine. Pattern Recognition Letters.

Tharwat, A., Hemedan, A.A., Hassanien, A.E., Thomas G. (2018). A biometric-based model for fish species classification. *Fisheries Research*, *204*, 324–336.

Tharwat, A., & Moemen, Y.S. (2017). Classification of toxicity effects of biotransformed hepatic drugs using whale optimized support vector machines. *Journal of Biomedical Informatics*, *68*, 132–149.

Wang, G., & Guo, L. (2013). A novel hybrid bat algorithm with harmony search for global numerical optimization. Journal of Applied Mathematics.

Wang, L. (2005). *Support vector machines: theory and applications, vol. 177*. Berlin: Springer.

Wu, C.H., Tzeng, G.H., Lin, R.H. (2009). A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. *Expert Systems with Applications*, *36*(3), 4725–4735.

Xi, M., & Sun, J. (2008). An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position. *Applied Mathematics and Computation*, *205*(2), 751–759.

Xinchao, Z. (2010). A perturbed particle swarm algorithm for numerical optimization. *Applied Soft Computing*, *10*(1), 119–124.

Yang, X.S. (2014). *Nature-inspired optimization algorithms*, 1st edn. Amsterdam: Elsevier.

Zhang, X., & Chen, X. (2010). An ACO-based algorithm for parameter optimization of support vector machines. *Expert Systems with Applications*, *37*(9), 6618–6628.

Zhang, Y., & Zhang, P. (2015). Machine training and parameter settings with social emotional optimization algorithm for support vector machine. *Pattern Recognition Letters*, *54*, 36–42.

Zhao, M., Fu, C., Ji, L., Tang, K. (2011). Feature selection and parameter optimization for support vector machines: a new approach based on genetic algorithm with feature chromosomes. *Expert Systems with Applications*, *38*(5), 5197–5204.