Soft Computing (2021) 25:3335–3352
https://doi.org/10.1007/s00500-020-05439-w

METHODOLOGIES AND APPLICATION

Evolutionary competitive swarm exploring optimal support vector machines and feature weighting

Ala' M. Al-Zoubi[1,2] · Mohammad A. Hassonah[2] · Ali Asghar Heidari[3,4] · Hossam Faris[2] · Majdi Mafarja[5] · Ibrahim Aljarah[2]

Published online: 2 January 2021

Abstract
To deal with classification problems, support vector machines (SVMs) are utilized in a wide variety of applications as effective and powerful supervised learning paradigm. However, the efficacy and outcomes of an SVM-based classification model is influenced by the proper selection of SVM parameters in addition to the nature of the datasets. Therefore, the purpose of this work is to enrich the efficacy of the SVMs based on simultaneous optimization of the parameters and feature weighting of these models. In this paper, an improved evolutionary variant of competitive swarm optimizer (CSO) is proposed to evolve the parameters of SVMs and optimize the weights of features. Simulations and experiments are performed based on various datasets from UCI repository to investigate the effectiveness of the proposed hybrid CSO-based SVM model versus genetic algorithm, particle swarm optimizer and the classical grid-based search. Results and analysis reveal that the proposed crossover-based mechanism inside CSO has improved the classification capabilities of the hybrid CSO-SVM technique.

Keywords Feature weighting · Support vector machines · Competitive swarm optimizer · Medical dataset

1 Introduction

The properties of a dataset and the parameter values of a certain classifier are two factors that highly influence the classifier's accuracy (Li et al. 2015; Huang and Wang 2006). In complex and high-dimensional datasets that are non-separable, kernel functions can be used to restructure the data in order to map it easily in higher dimensional spaces (Phan et al. 2017). However, there is no kernel function that is appropriate for all datasets, and identifying which kernel to be used with each dataset is a complex and time consuming process. Moreover, mapping the data into a new higher dimensional space increases the complexity of time and space resources (Zhang et al. 2010).

Therefore, many researchers tried to investigate working with datasets properties (e.g., number of features) rather than mapping them with higher dimensional spaces (Li et al. 2015; Al-Zoubi et al. 2019). Feature Selection (FS) and Feature Weighting (FW) are two of the most efficient approaches that have been used to tackle this problem. On one hand, FS methods focus on eliminating some irreverent and redundant features that may mislead the learning algorithms and, therefore, decrease their performance (e.g., accuracy). On the

Communicated by V. Loia.

✉ Ala' M. Al-Zoubi
alzoubi@correo.ugr.es; alaah14@gmail.com

Mohammad A. Hassonah
mohammad.a.hassonah@gmail.com

Ali Asghar Heidari
as_heidari@ut.ac.ir

Hossam Faris
hossam.faris@ju.edu.jo

Majdi Mafarja
mmafarja@birzeit.edu

Ibrahim Aljarah
i.aljarah@ju.edu.jo

[1] School of Science, Technology and Engineering, University of Granada, Granada, Spain

[2] King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

[3] School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran

[4] Department of Computer Science, School of Computing, National University of Singapore, Singapore, Singapore

[5] Department of Computer Science, Birzeit University, PO Box 14, West Bank, Palestine

other hand, FW methods tend to assign a rank (or a weight) for each feature based on its correlation with the class label, i.e., the highly correlated features will get a high weight (Tahir et al. 2007).

Support Victor Machine (SVM) is one of the most popular supervised machine learning techniques, which has been successfully applied to various classification problems such as text categorization, hand-written character recognition, image classification, and fault diagnosis (Shin et al. 2005; Yuan and Chu 2007; Guo et al. 2008; Wu et al. 2010; Sun et al. 2019; Tanveer 2015; Xu et al. 2013). Like other classifiers, the performance of SVM classifier is highly dependent on its parameter values and settings [e.g., penalty parameter C and the kernel function's parameters such as the gamma (γ) (Lin et al. 2008; Huang and Dun 2008)], in addition to the properties of the dataset being classified (e.g., what features to be used). Many researchers in the literature concentrated on optimizing the SVM parameters using different algorithms, while others used to select the most informative features in order to improve the performance of the SVM classifier. However, a few works that combines both problems are available. The Grid search is one of the common algorithms that have been used to set the C and γ parameters when using radial basis functions (RBF) kernel. However, using this method became impractical in terms of time complexity (Hsu et al. 2003; LaValle et al. 2004), and no attention is paid to which features to be used in the learning process.

Recently, many metaheuristic algorithms were employed to tune the SVM parameters and/or select the input subset of features. Some examples of those optimization methods are: Genetic Algorithm (GA) (Huang and Wang 2006), Whale Optimization Algorithm (WOA) (Ala'M et al. 2018) Multiverse optimization (MVO) (Sadiq et al. 2019), Simulated Annealing (SA) (Boardman and Trappenberg 2006), Salp Swarm Algorithm (Ala'M et al. 2020), and Particle Swarm Optimization (PSO) (Huang and Dun 2008). The Competitive Swarm Optimization (CSO) (Cheng and Jin 2015) is a recent optimization algorithm that was fundamentally inspired by the PSO with some modifications, where neither local best nor global best positions were used in CSO. In CSO, a competition mechanism is maintained between the particles of the swarm, where a pairwise competition is performed between the particles. The loser learns from the winner instead of learning from the *pbest* and *gbest* as in the original PSO. The CSO recorded competitive results in solving many optimizing problems especially the large-scale problems.

In this paper, an improved CSO is proposed to perform two main tasks simultaneously, which are the optimization of SVM parameters, and optimizing the weights of the input features. The new mechanism in CSO assists the algorithm in establishing a more stable balance between exploratory and exploitative tendencies by integrating a crossover mechanism into its procedure. Moreover, most of the previous works that combined metaheuristics with SVM and performed feature selection where the input features were either included or excluded. In this paper, the proposed approach automatically quantifies the weights of the input features and consequently gives an insight on the relative importance of these features. Therefore, it can provide a great aid for subject matter experts and decision makers in their domains. Therefore, it is expected that the proposed model is best suited for small to medium dimensional datasets. For verification, different experiments are conducted using popular medical datasets with different complexities. The proposed approach is compared with well-regarded algorithms in the literature that were commonly used in combination with SVM. The experimental simulations reveal that the superiority of the proposed CSO and its efficient performance compared to other methods.

The rest of this paper is organized as follows: In Sect. 3, preliminaries including SVM and CSO basics are discussed. The proposed approach is discussed in Sect. 4. In Sect. 5, the details of the experiments conducted in this paper are also discussed, followed with a deep analysis of the obtained results. Finally, the conclusion and future directions are drawn in Sect. 6.

2 Related works

Metaheuristic algorithms are known very well in the literature as high-level procedures designed to handle many complex problems that require searching, generating and selecting. In the literature, the application of metaheuristic algorithms in combination with SVM can be categorized into three main types.

The first type comprises the works designed to improve and optimize the hyper-parameters of the SVM, which are cost and gamma. The SVM usually suffers to reach the optimal solution due to the high range of the two parameters. Therefore, several works applied metaheuristic algorithms to solve this problem. For example, Friedrichs and Igel (2005) proposed an evolutionary approach to select and tune the SVM parameters, they applied the covariance matrix adaptation evolution strategy (CMA-ES) for this purpose. Additionally, they showed that their approach achieved better results than the standard grid search. Xiaofang and Yaonan (2008) utilized the chaos optimization algorithm to search for the optimal parameter values of the SVM. They argue that the chaos algorithm has an effective ability to reach the global optimal and eventually improve search efficiency and accuracy. The work in Lorena and De Carvalho (2008) investigated how improving and optimizing the SVM parameters can be enhanced for multiclass problems by applying GA. Another recent work Tharwat and Hassanien

(2019) employed Bat Optimization Algorithm (BA) alongside other metaheuristic algorithms to determine the best hyper-parameters and, thus, increasing the performance of the classification process. BA achieved competitive results when compared with other algorithms such as GA and PSO. Furthermore, many other works covered this area of research on tuning the SVM parameters (Guo et al. 2008; Zhang et al. 2010; Bao et al. 2013; Tharwat and Hassanien 2018). However, optimizing only the hyper-parameters is not effective in all problems, particularly in high-dimensional data.

This is when the second type comes, where this approach combines the optimized hyper-parameters with the feature selection simultaneously. Due to the emergence of high-dimensional data in recent years, the feature selection method has become required and important for such data (Zhao et al. 2014; Zhang et al. 2020). Simultaneously applying metaheuristic algorithms for the two tasks shows magnificent results as demonstrated in the literature. Faris et al. (2018) proposed an automatic approach to optimize the hyper-parameters as well as applying feature selection process to achieve the optimal result from the SVM. The authors used the Multi-Verse Optimizer (MVO) to achieve this; the MVO designed as a tuner to improve the main parameters of SVM and finding the best subset of features. Aljarah et al. (2018) also presented similar work by applying the recent algorithm Grasshopper Optimization Algorithm (GOA). The proposed approach was compared against seven well-known algorithms on 18 high- and low-dimensional benchmark datasets. The results show that their approach outperforms all seven algorithms in most of the datasets. In another research, a simultaneously feature subset selection and optimization of SVM parameters were applied to solve the multi-objective optimization problems by using the multi-objective Genetic Algorithm NSGA-II (Bouraoui et al. 2018). Similarly, several works were implemented using different metaheuristic algorithms including, Chaotic Ant Lion Optimization (CALO) (Tharwat and Hassanien 2018), PSO (Lin et al. 2008; Tu et al. 2007; Liu et al. 2011; Huang and Dun 2008), GA Huang and Wang (2006); Zhao et al. (2011); Reif et al. (2012), Ant Colony Optimization (ACO) (Huang 2009) and Gravitational Search Algorithm (GSA) (Li et al. 2015).

Similar to the other previous approaches, the final type consists of optimizing the hyper-parameters, however, differs in the way it deals with the features; as it performs feature weighting instead of selecting a subset of them. The weighting process operates by multiplying the value of every instance of all the features, then order them by their values. This method is considered more efficacious than the feature selection process in a number of cases and problems where the features are very sensitive. Thus, removing these kinds of features may negatively affect the classification performance. Phan et al. proposed a GA-SVM model that optimizes the hyper-parameters of SVM and weights the features to solve classification problems efficiently (Phan et al. 2017). The authors run their approach on several real-world datasets and compared them with the grid search and other standard search methods. The work introduced a good technique using the GA. To the best of our knowledge, Phan et al. was the earliest in this category. And since their work, there is no major advancements in this direction. Therefore, we conduct this research in order to boost the performance of SVM and show the capabilities of swarm intelligent algorithms in making a significant advancement in this line of research.

3 Preliminaries

3.1 Support vector machine

Support Vector Machines (SVM) is a well-established supervised machine learning technique designed for the first time by Vapnik (1999) to handle and find solutions for classification and regression scenarios, based on examining the given datasets and exploring definite hidden/visible patterns. SVM has shown its high capabilities in dealing with nonlinear classification problems. SVM method distributes the training data in other dimensional feature space, and then, it performs linear separation on the data for detecting possible classes (Wang and Chen 2020). SVM performs the separation step based on generating a hyper-plane, which is considered to optimize the observed margin among the nearest points of various classes, which are called support vectors (Shen et al. 2016). SVM tries to find the fittest hyper-plane as it is demonstrated in Fig. 1. However, this step has an inertia to over-fitting problem, which can lead to the misclassification of the new samples of datasets. To deal with this problem, this step is managed by using a penalty parameter cost (C) that increases the accuracy of classification and prediction.

The formula used in common SVM can be equipped with kernel functions to deal with the nonlinear separation problem, including polynomial kernel and the Radial Basis Function (RBF). RBF is the most popular kernel function utilized within SVM, which uses a Gamma (γ) parameter.

The most applied technique for searching and finding the parameters of SVM-based classifiers is grid search. Managing the misclassification error and obtaining the values of C and γ are simultaneously important to reach to high performance and avoid over-fitting drawback when using SVM with RBF kernels (Hsu et al. 2003). However, because grid search technique performs local search process, it has the tendency to be stagnated to local optima (LO) intervals. In this method, if we set small searching intervals, we will obtain poor results and if we set large intervals, it will bring more computational time for the operating system (LaValle et al. 2004; Hsu and Lin 2002). Additionally, this algorithm cannot be used for Feature Selection (FS).

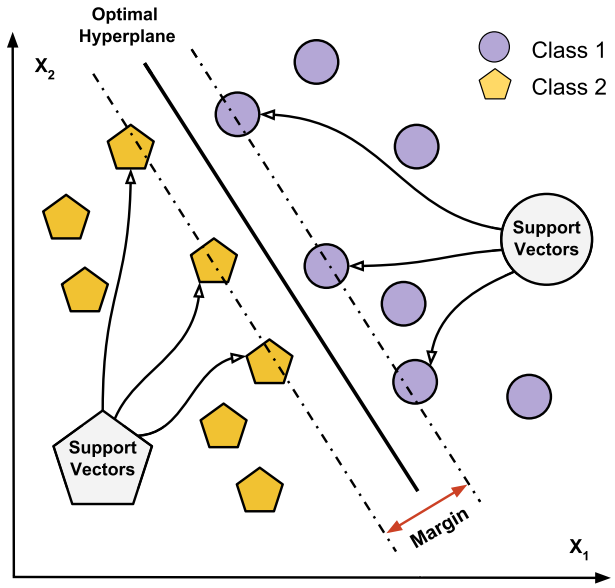


Fig. 1 Support vectors and optimal hyperplane in support vector machines

3.2 Competitive swarm optimization

CSO is known as a successful variant of PSO technique (Cheng and Jin 2015). The well-known PSO cannot show an excellent performance when dealing with high-dimension problems, as reports and observations confirmed, because of many local best positions we may face in these cases (Yang and Pedersen 1997). When PSO cannot manage a fine balance between exploration and exploitation, the problem of premature convergence can happen, which is a frequently observed drawback in metaheuristic approaches including PSO (Chen et al. 2013). To cope with immature convergence trends and tendency to stagnation drawbacks, many variants of PSO were introduced in recent years (Cheng and Jin 2015). However, the majority of these variants still may face the stagnation dilemma, because the global best search particle (*gbest*) has a significant impact on the efficacy of PSO. In CSO, it was intended to decrease the impact of *gbest* in order to alleviate the problems of convergence.

The structural and conceptual difference of CSO with PSO has two points. First, in PSO technique, exploration and exploitation mechanisms are motivated by *gbest* and the personal best particle *pbest*, whereas in CSO, there is no *pbest* and *gbest* within the optimization steps, and the exploratory and exploitative behaviors are observed based on the pairwise competition of particles. In this way, any particle has a chance to be a leader. Second, in CSO, there is no history, and when the particles lose the competition, they try to learn only from the winner particles in the ongoing set of solutions. In CSO, the set of search particles at iteration t $P(t)$ is obtained

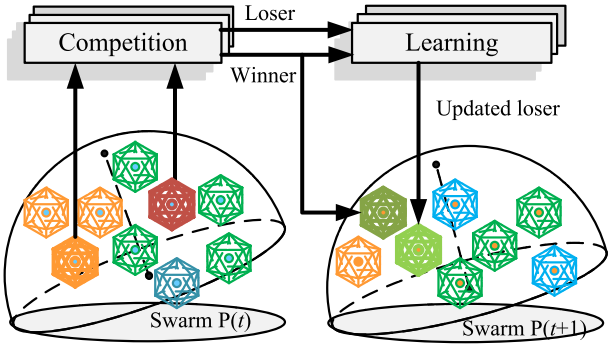


Fig. 2 CSO competition and learning process

based on:

$$P(t) = \{X_1(t), \ldots, X_m(t)\} \tag{1}$$

$$X_i(t) = \{x_{i1}, \ldots, x_{id}(t)\} \tag{2}$$

where X shows a d-dimensional particle $X \in R^d$ and m is population size. In CSO, we need to randomly divide the initial set of particles into two groups with size of $m/2$. A base particle (one particle from the first group) is considered to compete with the paired one in the other group. The winner is the particle with better fitness, and we directly transfer it into the next step to be used in $P(t+1)$, while the other one is the looser and should learn from winner, that is to update its velocity and position vectors with regard to those for winner, to be inserted to the next iteration. By this manner, $m/2$ competitions are performed to update half of the population. Fifty percent of particles are the winners; hence, they are inserted directly to the next level, and the rest of them are treated as looser particles, which will be passed after updating of their status, as shown in Fig. 2.

Each particle has vectors for position and velocity. In i-th pairwise race in the t-th iteration, the position and velocity vectors of winner and loser particles are denoted by $X_{w,i}(t)$, $X_{l,i}(t)$, $V_{w,i}(t)$, $V_{l,i}(t)$, respectively, while we have $i = 1, 2, \ldots, m/2$. After i-th race, the loser is updated based on:

$$\begin{aligned} V_{l,i}(t+1) = {} & R_1(i,t)V_{l,i}(t) + R_2(i,t)(X_{w,i}(t) - X_{l,i}(t)) \\ & + \varphi R_3(i,t)(\bar{X}_i(t) - X_{l,i}(t)) \end{aligned} \tag{3}$$

$$X_{l,i}(t+1) = X_{l,i}(t) + V_{l,i}(t+1) \tag{4}$$

where $R_1(i,t)$, $R_2(i,t)$, $R_3(i,t) \in [0,1]^d$ denote random vectors, $\bar{X}_i(t)$ represents the average locations of the some particles, where, we can calculate the average of locations for ongoing swarm or using predefined neighboring particles. φ is the single tunable factor of CSO to manage the impact of $\bar{X}_i(t)$. The pseudocode of CSO is represented in Algorithm 1.

Algorithm 1 Pseudocode of CSO

```
Inputs: Number of iterations L, population size m
output: The best particle and its fitness value
Initialize P(0) in a random manner
while t < L do
    Obtain the fitness (F) of particles P(t)
    U = P(t) ▷ Set of particles that still have not considered in race
    P(t + 1) = ϕ
    while U ≠ ϕ do
        Select two random particles X1(t), X2(t) from U
        if F(X1(t)) < F(X2(t)) then
            Xw(t) = X1(t); Xl(t) = X2(t);
        else
            Xw(t) = X2(t); Xl(t) = X1(t);
        P(t + 1) = P(t + 1) ∪ Xw(t)
        Update Xl(t) using Eqs. (3) and (4) to get Xl(t + 1)
        P(t + 1) = P(t + 1) ∪ Xl(t + 1)
        U = U − (X1(t) ∪ X2(t))
    t = t + 1
```

4 Proposed approach

In this section, we first describe the modified version of CSO (CCSO) which introduces a crossover operator embedded in it. Then, we describe, in detail, the proposed CCSO-SVM classification model which deploys CCSO for optimizing the parameters of SVM and to perform feature weighting simultaneously.

4.1 CSO with crossover (CCSO)

To enhance the efficacy of CSO, we utilize a crossover procedure to combine the particles, as described in Eq. 5.

$$X_l(t) = \texttt{Crossover}(X_l(t), X_{\text{best}}) \tag{5}$$

where `Crossover()` is a new process that executes the crossover-based process on loser particles, $X_l(t)$ is the loser particle of the tth iteration, and X_{best} is the particle with the best fitness out of all iterations.

After a pairwise competition is conducted between winner and loser particles where the loser particles are updated according to the winners, the crossover mechanism takes its place. Loser particles are utilized in the crossover operation between them and the best particle found so far. This technique guarantees further exploration and enhancements on the loser particles by making similar random chances of either selecting a number of positions from the best particle or keeping current positions, according to the following equation:

$$X_l^i = \begin{cases} X_{\text{best}}^i & \text{rand}(i) = 1 \\ X_l^i & \text{rand}(i) = 0 \end{cases} \tag{6}$$

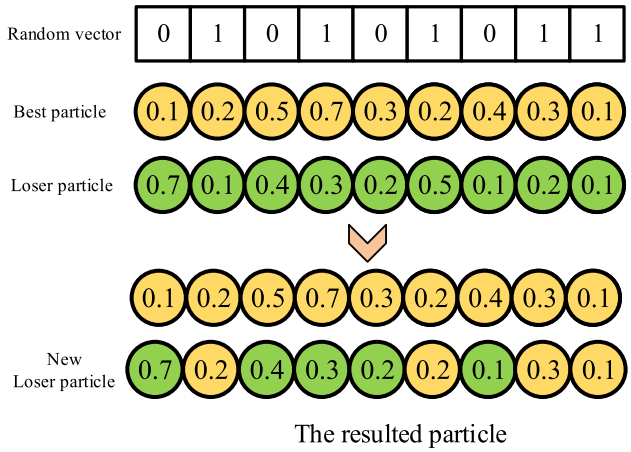


Fig. 3 The used crossover scheme

where X_l^i is the ith position in the loser particle, X_{best}^i is the ith position in the best particle, and $rand(i)$ is the corresponding random element.

Algorithm 2 Pseudocode of `Crossover()` function

```
Inputs: dimension dim, maximum iteration L, current iteration t,
    loser particle Xl(t), best solution Xbest
output: a new particle
Define rand = rand(1,dim);
if rand >= (t/L) then
    rand=1;
else
    rand=0;
for i = 1, i < size(Xl(t)), i++ do
    if (rand(i) == 1) then
        X_l^i(t) = X_best^i
```

An illustration of this operation is demonstrated in Fig. 3.

As shown in Fig. 3, we see that this operator exchanges the values between two particles. By this rule, we can experience sudden fluctuations in the loser particle. This operator tries to generate an intermediate particle within the feature space to assist CSO in exploring the feature weights.

Algorithm 3 explains the new steps for CSO after embedding the crossover technique. As the algorithm shows, the fitness of the updated particle X_{l_cross} is compared against the fitness of both X_{best} and X_l, and the particle with the best fitness is moved to the next iteration.

4.2 CCSO-SVM classification model

Before applying the CCSO algorithm for performing the tasks of optimizing SVM parameters and optimizing the weights of the input features, the representation of the solution (also known as particle or individual), and the selection of the fitness function should be resolved. In the following, we discuss each of these important design issues then

Algorithm 3 Pseudocode of CCSO

Inputs: Number of iterations L, population size m
output: The best particle and its fitness value
Initialize $P(0)$ in a random manner
Initialize $X_{best} = 1e200$
while $t < L$ **do**
  Obtain the fitness (F) of particles $P(t)$
  $U = P(t)$ ▷ Set of particles that still have not considered in race
  $P(t+1) = \phi$
  while $U \neq \phi$ **do**
    Select two random particles $X_1(t)$, $X_2(t)$ from U
    if $F(X_1(t)) < F(X_2(t))$ **then**
      $X_w(t) = X_1(t)$; $X_l(t) = X_2(t)$;
    else
      $X_w(t) = X_2(t)$; $X_l(t) = X_1(t)$;
    $P(t+1) = P(t+1) \cup X_w(t)$
    Update $X_l(t)$ using Eqs. (3) and (4) to get $X_l(t+1)$
    $X_{l\_cross}(t)$ = `Crossover(`$X_l(t)$`,`X_{best}`);` ▷ Call Algorithm 2
    if $F(X_l(t)) < F(X_{best})$ **then**
      $X_{best} = X_l(t)$;
    if $F(X_{l\_cross}(t)) < F(X_{best})$ **then**
      $X_{best} = X_{l\_cross}(t)$;
    $X_l(t+1) = X_{best}$;
    $P(t+1) = P(t+1) \cup X_l(t+1)$;
    $U = U - (X_1(t) \cup X_2(t))$;
  $t = t + 1$

we describe the overall system architecture of the proposed model.

4.2.1 Solution representation

As a search algorithm designed for solving sophisticated problems, CCSO is simultaneously utilized in two parts. The first part comprises searching for the best C and γ parameters for SVM classifier, and the second part includes weighting the features (see Fig. 5). Therefore, the number of elements generated by CCSO covers both parameters in addition to D number of features for every dataset, all combined in one-dimensional vector of $D + 2$ real numbers originally generated in the interval [0,1].

The first two elements in the vector are correspondent to C and γ parameters. The search spaces for these parameters are different from the original scale and, therefore, they are scaled to the intervals [0,35000] for C and [0,32] for γ. This transformation is performed using Min-max normalization as given in Eq. 7.

$$B = \frac{A - \min_A}{\max_A - \min_A}(\max_B - \min_B) + \min_B \tag{7}$$

The remaining elements that correspond to the features will be at the same original scale to be used for weighting. In the weighting process, each element in the vector is multiplied by the value of its matching feature for every instance 4. For example, if we have a simple dataset of 3 instances, then the values of the first feature of the 3 instances will be multiplied by the value of the first element of the solution generated by the CCSO. The same is applied for the rest of dataset as shown in Fig. 4. The overall solution structure and its presentation is illustrated in Fig. 5.

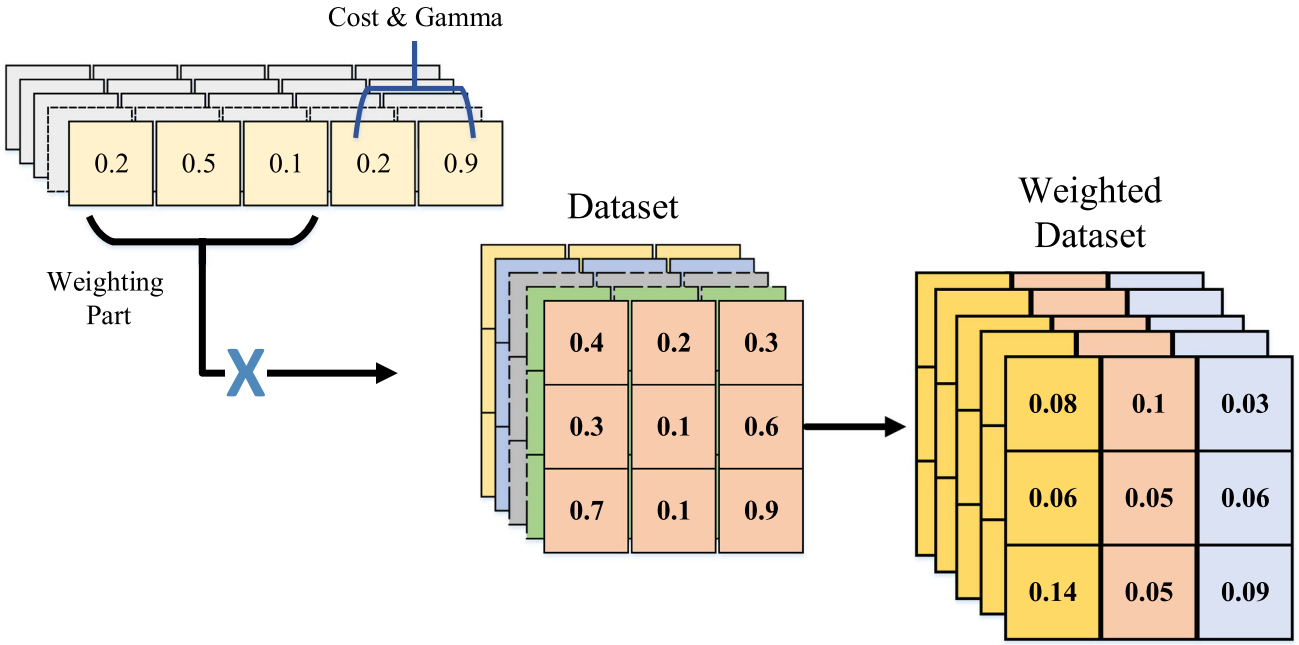


Fig. 4 A simple example to illustrate the representation of the weighting mechanism in the solution of the proposed model

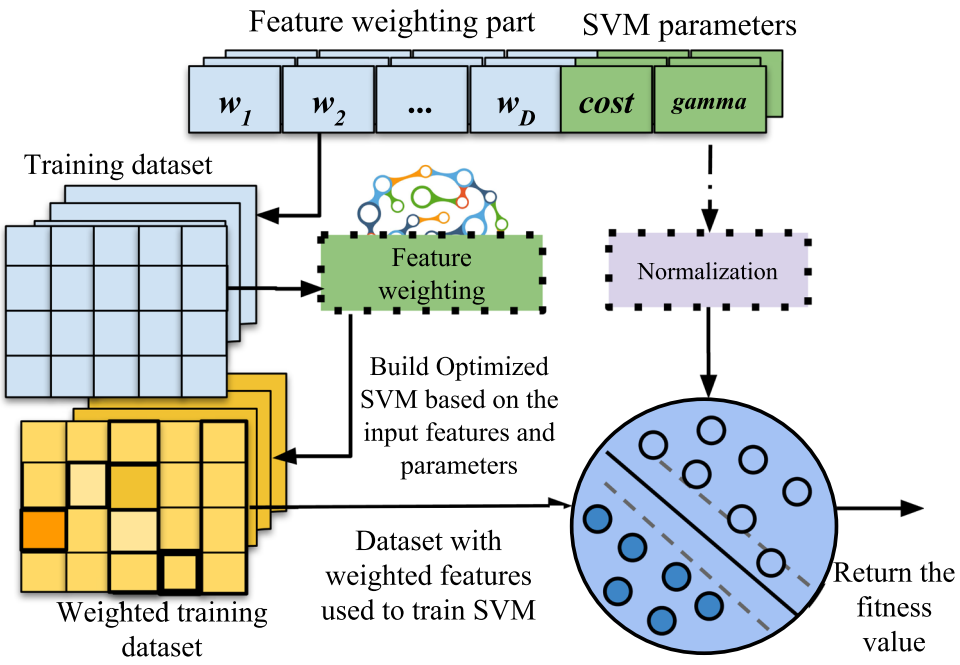


Fig. 5 Structure of the solution in the proposed CCSO-SVM model

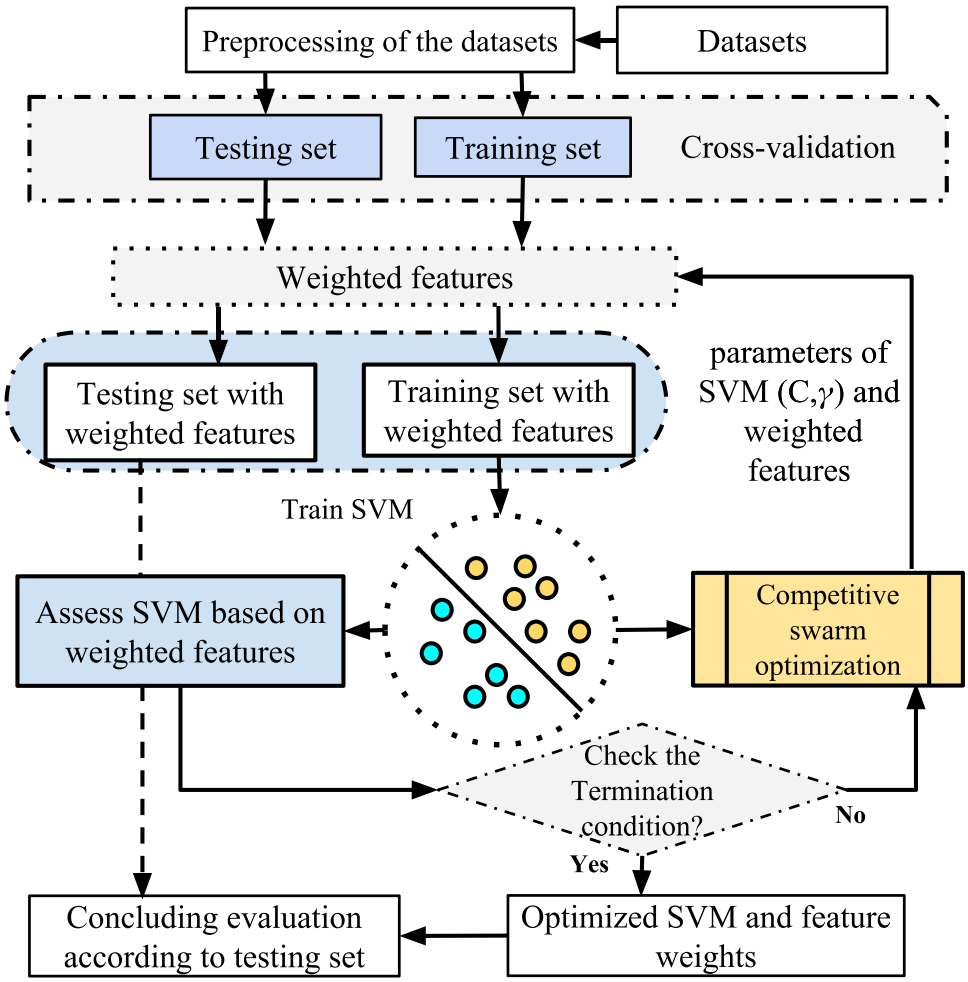


Fig. 6 The proposed CCSO-based process

Table 1 List of used datasets

#	Dataset	# Features	# Instances	# Classes	Class ratio
1	Blood	4	748	2	0.31
2	Colon cancer	2000	62	2	0.50
3	Diabetes	8	768	2	0.54
4	Haberman	3	306	2	0.32
5	Heart	13	270	2	0.80
6	Libras	90	360	15	1.00
7	Liver	6	345	2	0.73
8	Parkinsons	22	195	2	0.33
9	Spectf	44	267	2	0.26
10	WDBC	30	569	2	0.59

4.2.2 Fitness evaluation

The assessment of each individual in every iteration is performed by using the fitness function to provide the feedback for CSO and CCSO. The fitness function that is chosen for evaluation is the classification accuracy of the SVM classifier, which is calculated according to the following equation:

$$\text{fitness}(I_i^t) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N}\sum_{j=1}^{N}\delta(c(x_j), y_j) \tag{8}$$

where $c(x_j)$ is the accuracy of the jth instance of the testing set, y_j is the label of the actual class for the jth instance, δ denotes the relation between $c(x_j)$ and y_j, i.e., if $c(x_j) = y_j$, then $\delta = 1$, otherwise $\delta = 0$. The number of instances in the testing set is denoted by N, and K is number of folds.

4.2.3 System architecture

The processes that are carried out to fulfill our proposed approach start with splitting every dataset into training and testing sets. The splitting criterion depends on the number of separate experiments. In other words, for k experiments, the dataset is split into k parts, $k - (1/k)$ parts are used for training, and the remaining $1/k$ part is used for testing. This guarantees maximum diversity of both training and testing sets to produce the best possible model.

The next step includes involving the CSO and the proposed CCSO algorithms. In this step, CCSO starts its iterations with a randomly generated vector of real numbers, which is then used for setting C, γ and the weights of the features. Then, the SVM classifier starts training using the weighted training set. During the training process, an inner cross-validation is used in order to produce a more robust model.

After finishing the training process, SVM classifier returns the accuracy as the fitness value to the CCSO algorithm. The previous processes are repeated for the same training set until the termination criterion for CCSO is met which, in our case, is the maximum number of iterations.

Table 2 The detailed settings of the utilized system

Name	Settings
Hardware	
CPU	Intel Core (TM) i5-6400 processor
Frequency	2.70GHz
RAM	8 GB
Hard drive	500 GB
Software	
Operating system	Windows 7 (64-bit)
Language	MATLAB R2016a

Table 3 The detail of runs

Item	Settings
Splitting criteria	Tenolds
Population size	30
Iterations	50

Table 4 The parameter settings

Algorithm	Parameter	Value
GA	Single point crossover	1
	Mutation	0.01
	Roulette wheel selection	
PSO	Topology fully connected	
	Inertia factor	0.3
	c_1	1
	c_2	1
CSO	Phi	0.2
CCSO	Phi	0.2
	Crossover	1

Table 5 Different population × iteration

Dataset	10 × 30	30 × 50	50 × 100
Haberman	71.5054	**74.8602**	72.5376
Heart	84.4444	**85.5556**	84.0741

Bold indicate the best results

Table 6 Different Phi parameters of the CCSO

Dataset	0.2	0.4	0.6	0.8
Haberman	**74.8602**	72.2043	73.2151	72.9355
Heart	**85.5556**	81.8519	82.2222	82.9630

Bold indicate the best results

When the maximum number of iterations is reached, the best individual produced by CCSO is used for testing using the testing set. Finally, all previous steps are repeated for k times and the average values are considered. Figure 6 depicts all the previous processes.

5 Experiments and results

In this section, all results and experiments are reported, in detail, to show the performance of different algorithms in dealing with different datasets, in addition of a brief description regarding the importance of feature weighting.

5.1 Experimental setup and parameters tuning

In this work, in order to have a fair comparison, all tests are experienced based on the same conditions. We also used a personal computer with the specifications reported in Table 2. To investigate the performance of the proposed CCSO-SVM, ten well-known datasets are utilized from the UCI repository (Lichman 2013). Table 1 shows the details of the selected datasets. It is noteworthy to mention that, the class ratio is calculated by dividing the number of instances of the minor class by the number of the major instances.

Details of tests and runs are shown in Table 3. The parameter settings of the CCSO, CSO, GA, and PSO are shown in Table 4. To set the population size and number of iterations, different combination of these two parameters are conducted for CCSO which are 10 × 30, 30 × 50, and 50 × 100, respectively. Two datasets are selected for these initial experiments as they show high sensitivity during building the models in terms of accuracy results. As shown in Table 5, 30 for population size and 50 iterations produce the best accuracy. Furthermore, different values for Phi are tested as well. Table 6 shows that the best results for CSSO are obtained when Phi is 0.2.

Table 8 P values of CCSO versus other metaheuristic methods and grid search

Dataset	CSO-SVM	GA-SVM	PSO-SVM	Grid-SVM
Blood	7.13E−01	8.07E−01	2.70E−01	9.80E−03
Colon cancer	2.88E−04	2.50E−03	9.53E−05	1.93E−11
Diabetes	1.99E−02	9.03E−01	3.91E-−01	3.90E−01
Haberman	9.03E−01	3.56E−09	2.29E−06	5.37E−04
Heart	1.05E−01	8.30E−03	3.24E−02	3.71E−05
Libras	1.04E−04	1.17E−05	3.36E−02	4.59E−24
Liver	1.76E−01	1.93E−02	2.06E−04	3.60E−04
Parkinsons	5.90E−01	1.50E−03	1.12E−05	1.24E−17
Spectf	3.35E−04	3.63E−05	8.47E−02	1.10E−03
WDBC	6.07E−01	3.09E−01	5.20E−03	1.31E−01

Table 7 Comparison of CCSO and CSO with other metaheuristic methods and grid search in terms of accuracy rates

Dataset	CCSO-SVM		CSO-SVM		GA-SVM		PSO-SVM		Grid-SVM	
	Acc	Std	Acc	Std	Acc	Std	Acc	Std	Acc	Std
Blood	78.4721	5.3339	**79.2775**	3.1610	78.3441	5.0909	78.3459	6.0153	76.5964	5.0248
Colon cancer	**85.4762**	16.5510	84.0476	6.8319	82.3810	11.7824	80.4762	10.7539	64.5238	24.9578
Diabetes	76.9498	5.9196	**77.6179**	4.9922	76.6866	4.7283	76.9498	4.3139	76.2867	6.1448
Haberman	**74.8602**	10.0251	73.5484	9.3122	70.2473	8.8573	71.8925	5.8420	72.2473	6.9439
Heart	**85.5556**	6.1605	84.0741	7.4177	83.3333	5.8561	83.7037	7.0273	81.4815	5.2378
Libras	**89.7222**	1.8749	88.8889	6.1419	86.9444	5.2460	86.6667	7.7247	78.6111	9.6270
Liver	71.3277	6.8961	71.3277	9.9362	**74.2353**	7.7316	68.4034	5.5008	68.1345	7.8961
Parkinsons	**95.9211**	4.0432	94.8947	5.8781	92.3158	7.7161	93.8421	4.6237	86.1579	8.2616
Spectf	**82.3789**	5.3805	79.4160	4.3047	78.6752	5.7696	80.1994	10.4448	79.4302	6.5003
WDBC	**97.8916**	1.6114	97.7162	2.1951	97.0081	3.1078	97.3653	1.4889	97.3653	2.2255

Bold indicate the best results

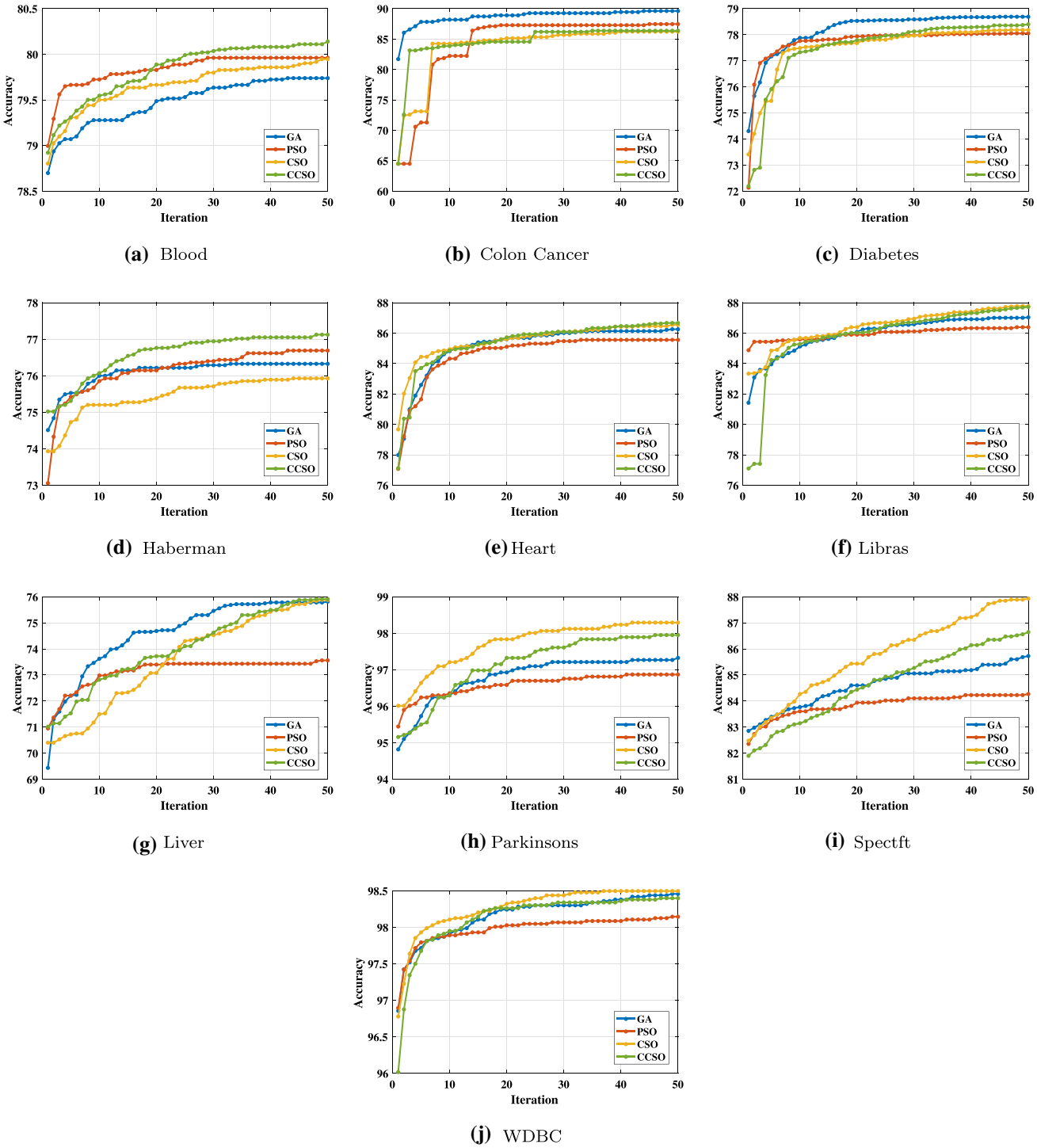


Fig. 7 Convergence curves of CCSO, CSO, GA, and PSO techniques

5.2 Comparison with other well-regarded metaheuristic-based SVM

Table 7 compares the accuracy results of the proposed CCSO against CSO and other peers. As per results in Table 7, it is observed that the proposed CCSO outperforms other competitors in terms of accuracy rates on Colon cancer, Haberman, Heart, Libras, Parkinsons, spectf, and WDBC datasets (70% of datasets). Compared to Grid-SVM, we see that all swarm-based optimizers show a better performance in terms of average (Acc) and (Std) results. The CSO reveals the best results on Blood and Diabetes cases, while GA obtains

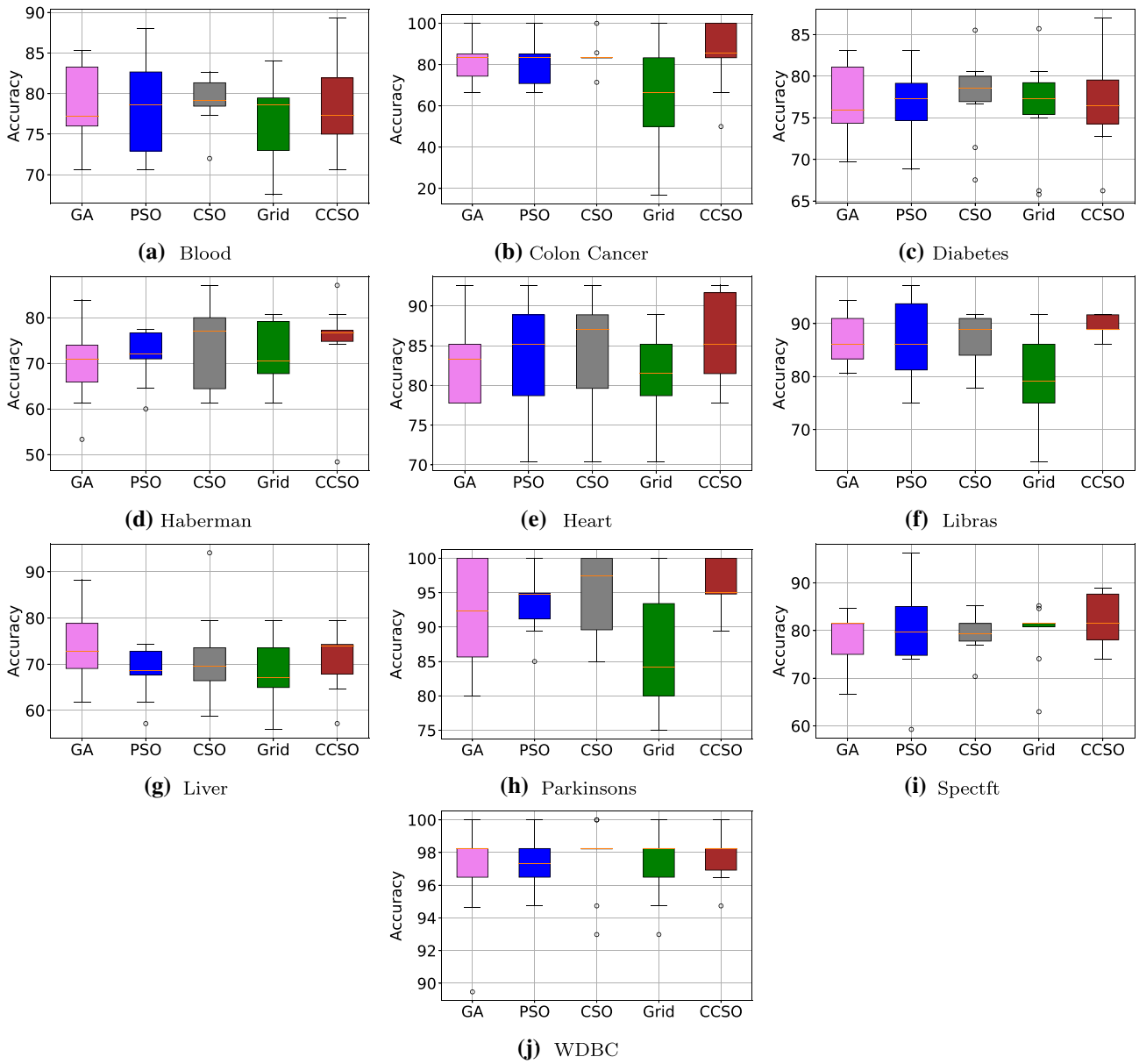
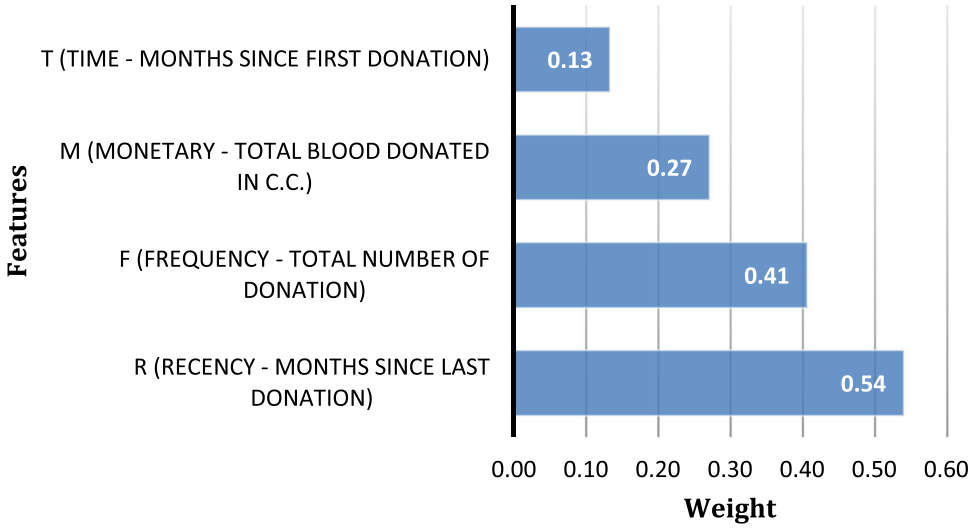


Fig. 8 BoxPlot results for CCSO, CSO, GA and PSO

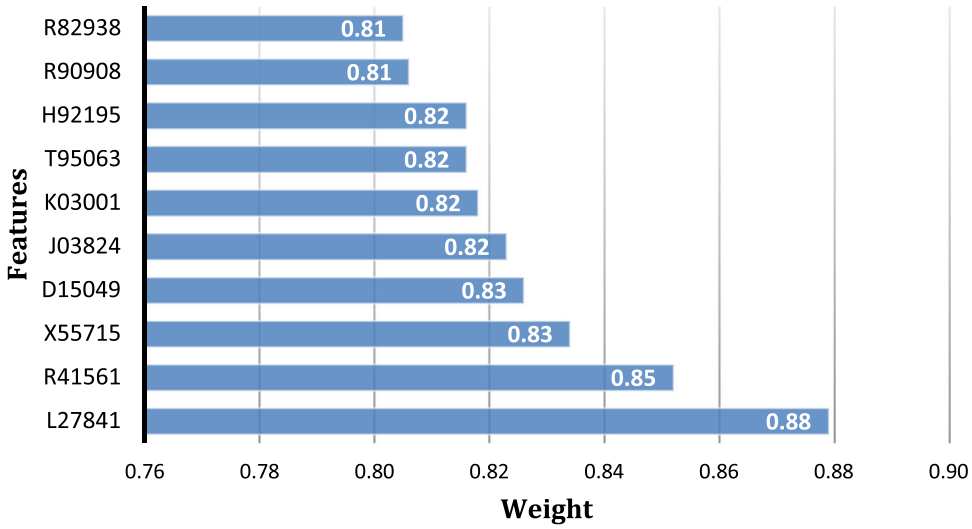
the highest rate on Liver case. The CCSO shows the highest accuracy results for a large dataset like Colon cancer and it improves the results of CSO, GA, PSO, and Grid search up to, 1.428%, 3.095%, 5%, and 20.952%, respectively. Based on overall ranks, we can see that the CCSO is ranked one, followed by CSO, GA, PSO, and Grid-search methods. The main reason is that the CCSO has an improved capability in balancing the exploratory and exploitative trends when dealing with more complex feature spaces. The integrated crossover scheme has assisted CCSO to show a more efficient performance in terms of local optima escaping behaviors compared to other peers.

The nonparametric statistical test Wilcoxons rank sum is conducted to test the significance of the obtain results of CCSO against the other metaheuristic algorithms. In this work, the test is performed at 5% significance level. Table 8 shows the statistical test results in terms of p values. Note that the p values that are less than 0.05 which indicate a significance difference are underlined. As per the obtained p values, we see the differences are significant for most of the cases. That is CCSO significantly outperforms GA in 6 datasets, and it outperform PSO, and grid search in 7 datasets.

The convergence curves of CCSO are monitored and compared with CSO, PSO, and GA in Fig. 7. As per curves, first, we see the curves of CSO-based approach are superior to



(a) Blood



(b) Colon Cancer

Fig. 9 Features weights obtained by CCSO method for blood and colon cancer datasets

PSO and GA in most of the datasets. In addition, we see the curves of CCSO and CSO are very competitive for some datasets such as Colon Cancer, Diabetes, Heart, and WDBC cases. For some of them such as Haberman and Heart, GA and PSO also show a competitive efficacy in convergence trends. We observe that CCSO has a better capacity to avoid local optima stagnation drawbacks; hence we see it outperforms basic CSO in dealing with Blood, Haberman, Heart, and Liver datasets. However, the basic CSO also is ranked one on two cases: Parkinsons and Spectf cases. In overall, we see the CCSO can show improved convergence leanings due to its higher exploration capacities in initial steps and its enriched capabilities in performing a smoother transition from diversification to intensification in the last steps. The crossover between the loser particles and the leaders has enhanced the quality of the swarm in a gradual manner (Figs. 7, 8).

In order to further exhibit the distribution of the accuracy rates of CCSO-SVM versus the other optimized SVMs and to study the stability of the algorithms, we show the boxplots of the obtained results in Fig. 9. The boxplot results also confirm the satisfactory efficacy of the proposed CCSO-SVM as it shows very competitive stability compared to the other algorithms in most of cases as it has relatively smaller boxes.

The resulted features' weights calculated by CCSO method are shown in Figs. 9, 10, 11, 12 and 13. Note that in case there are more than 10 input features in the dataset, only the ten features that have the highest weights are shown for clarity reason. The weighting method is a meaningful

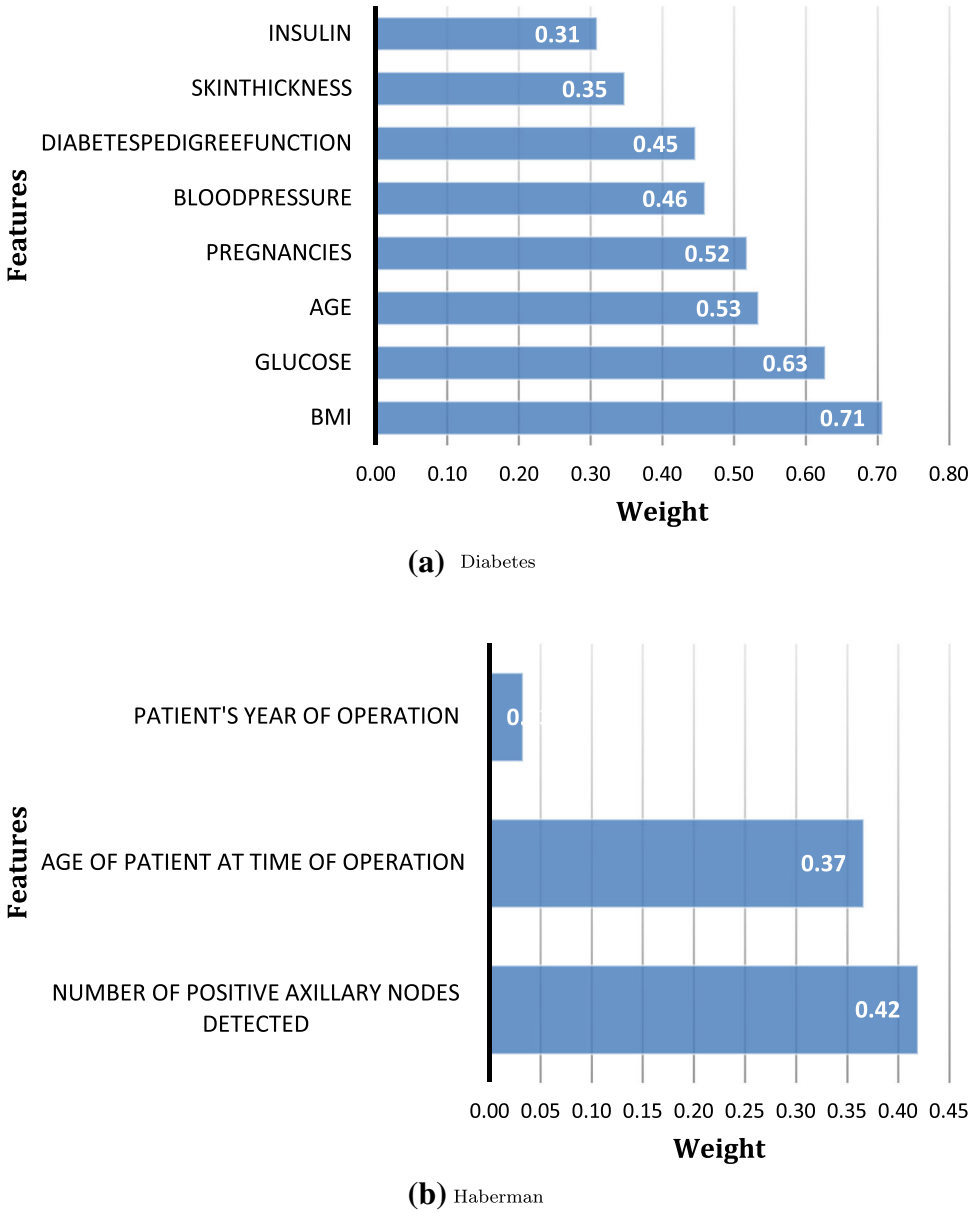
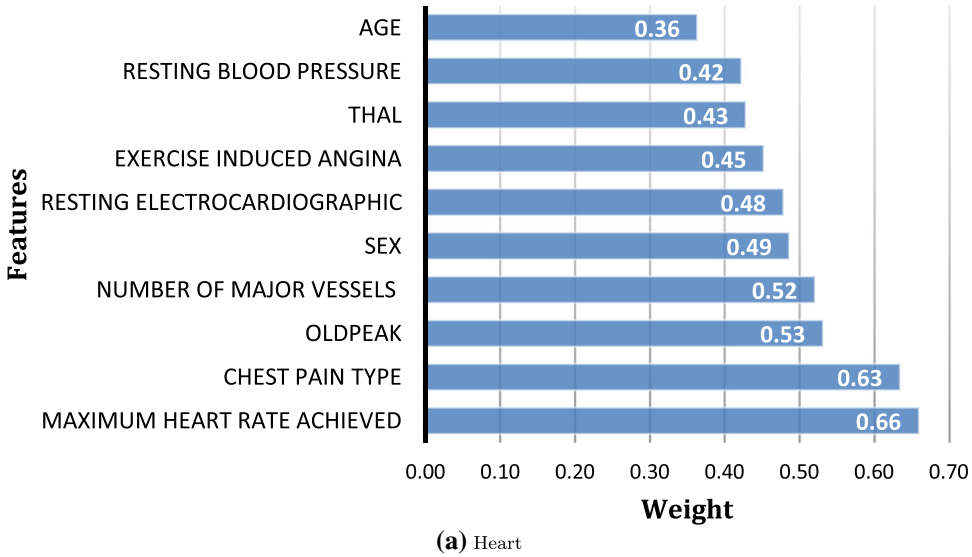


Fig. 10 Features weights obtained by CCSO method for diabetes and Haberman datasets

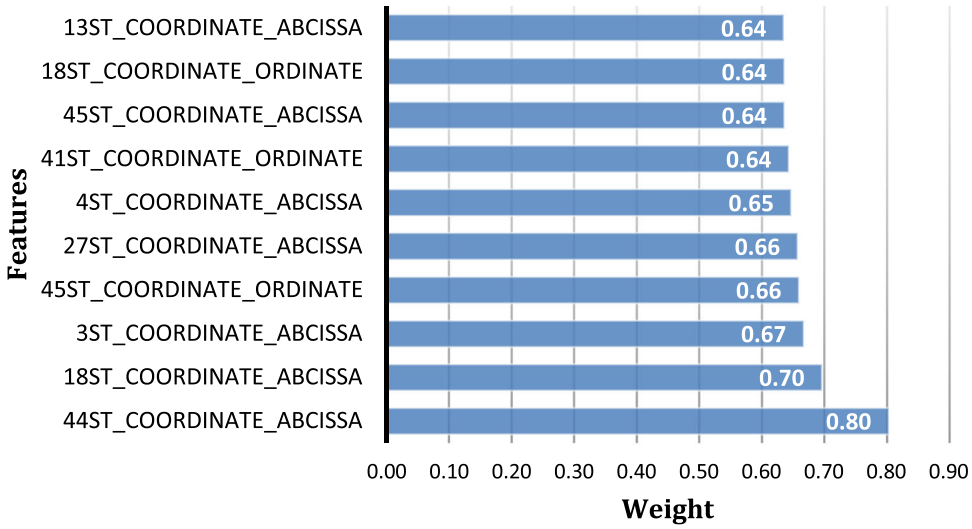
way to check the most influencing features for every dataset. As it can be seen in the figures, the proposed CCSO-SVM automatically identifies the weights of the input features for each dataset.

For example, as shown in the Heart dataset figure (Fig. 11), the features have been weighted via CCSO algorithm and arranged by their importance of detecting the status of the heart patient, whether the disease is present or absent. The *Age* feature, for instance, is the least important among all features, this indicates according to the developed model that the age has less effect for such disease, unlike another feature such as *Sex* which appears to be more important for the detection process of the heart disease. In addition, the *maximum heart rate achieved* feature, which is the maximum number of times the heart should beat per minute, shows that the heart rate is essential for the detection process, where the normal adult human heart rate should be between 60 and 100 bpm while resting according to The American Heart Association.

In this work, we select the medical field as an application to form an essential guide for concerned parties in that field. Such guidance can also be implemented to seek the pattern and hidden information of the selected features, such as the most and least important feature. This method will help in preventing the use of useless and time-consuming features and counting on the important ones. In addition, these weights could help decision makers to have better knowledge about the key factors in identifying a specific disease.



(a) Heart



(b) Libras

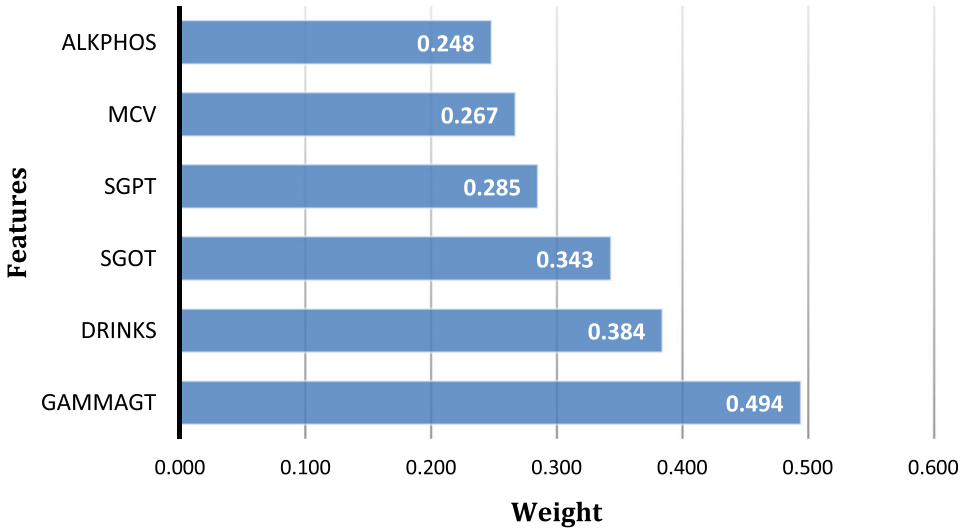
Fig. 11 Features weights obtained by CCSO method for heart and Libras datasets

5.3 Comparison with other classification algorithms

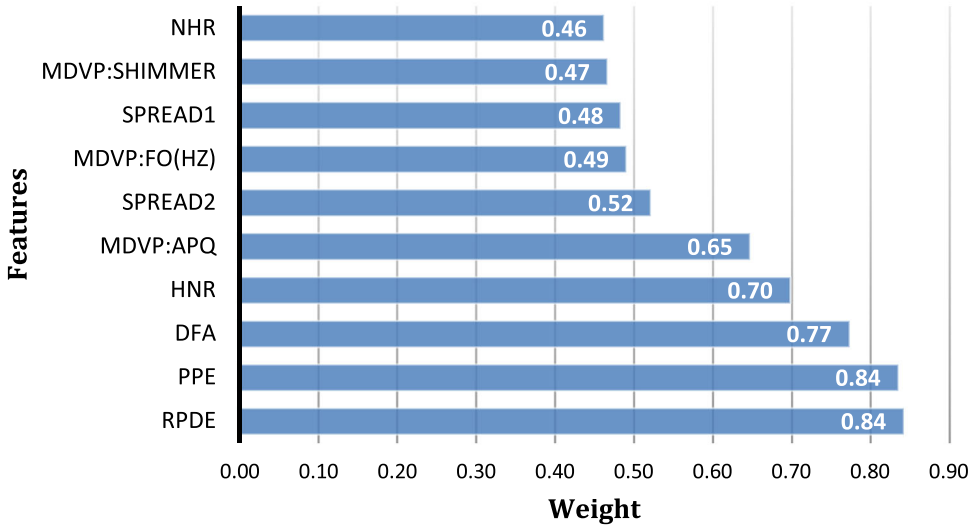
In this experiment, we compare the performance of the proposed CCSO-SVM with well-regarded classification algorithms that are widely used in the literature. These classifiers include: decision trees algorithm C4.5, k-nearest neighbor (k-NN), Naïve Bayes (NB), Multi-layer Perceptron (MLP) neural network. For C4.5, we used its Java implementation in Weka which is known as (J48). For k-NN, k was set to 1 which gave the best performance. In MLP, the number of hidden neurons was set according to the popular rule which determines this number as the average of input features and number of outputs. Table 9 tabulates the results of all these classifiers. It can be clearly seen that CCSO-SVM obtained the best results in majority of the datasets (7 out 10 datasets). While MLP obtained the best results in two cases, which are Liver and Spectf datasets. Finally, k-NN achieved the best results in one case only, which is Parkinsons, with a slightly better result than CCSO-SVM.

5.4 Comparison with other methods in the literature

Table 10 shows the results of the CCSO-SVM and other proposed approaches in the literature, namely, CSO-ELM & CSO-RELM (Eshtay et al. 2018), TMGWO2 (Abdel-Basset et al. 2020), GCACO (Moradi and Rostami 2015), and UPFS (Dadaneh et al. 2016). It can be seen that the CCSO-SVM outperforms all the other approaches in five datasets and achieved competitive results with the rest of the datasets. Therefore, the results of this section proved the superiority of the proposed method.



(a) Liver



(b) Parkinsons

Fig. 12 Features weights obtained by CCSO method for Liver and Parkinsons datasets

As any metaheuristic algorithm, the complexity of the proposed wrapper method highly depends on the complexity of the fitness function. In our case, this can be expressed by the complexity of the SVM with RBF kernel which is $O(n_{sv}d)$ where n_{sv} denotes the number of support vectors and d represents the number of input dimensions.

6 Conclusion and future directions

In this work, we proposed an improved CSO-based hybrid SVM model that enhances the accuracy results of SVM based on the exploration and exploitation mechanisms of a crossover-based CSO technique. The developed system and model are utilized to optimize the parameters of SVM in addition to feature weights in dealing with several classification cases. The proposed hybrid CCSO-based SVM model was compared with CSO, GA, PSO and Grid search methods in terms of accuracy results and convergence behaviors. The statistical results on ten datasets from UCI show that the proposed approach can obtain satisfactory results and the best parameters and attribute weights.

For future works, the developed CCSO-based SVM model can be utilized to deal with different applications in medical diagnosis and geoscience areas, such as remote sensing datasets. In addition, a multi-objective version of the proposed CCSO can be utilized to deal with different objective functions, simultaneously. Parallel computing is another direction that is worth investigating in order to speed up the

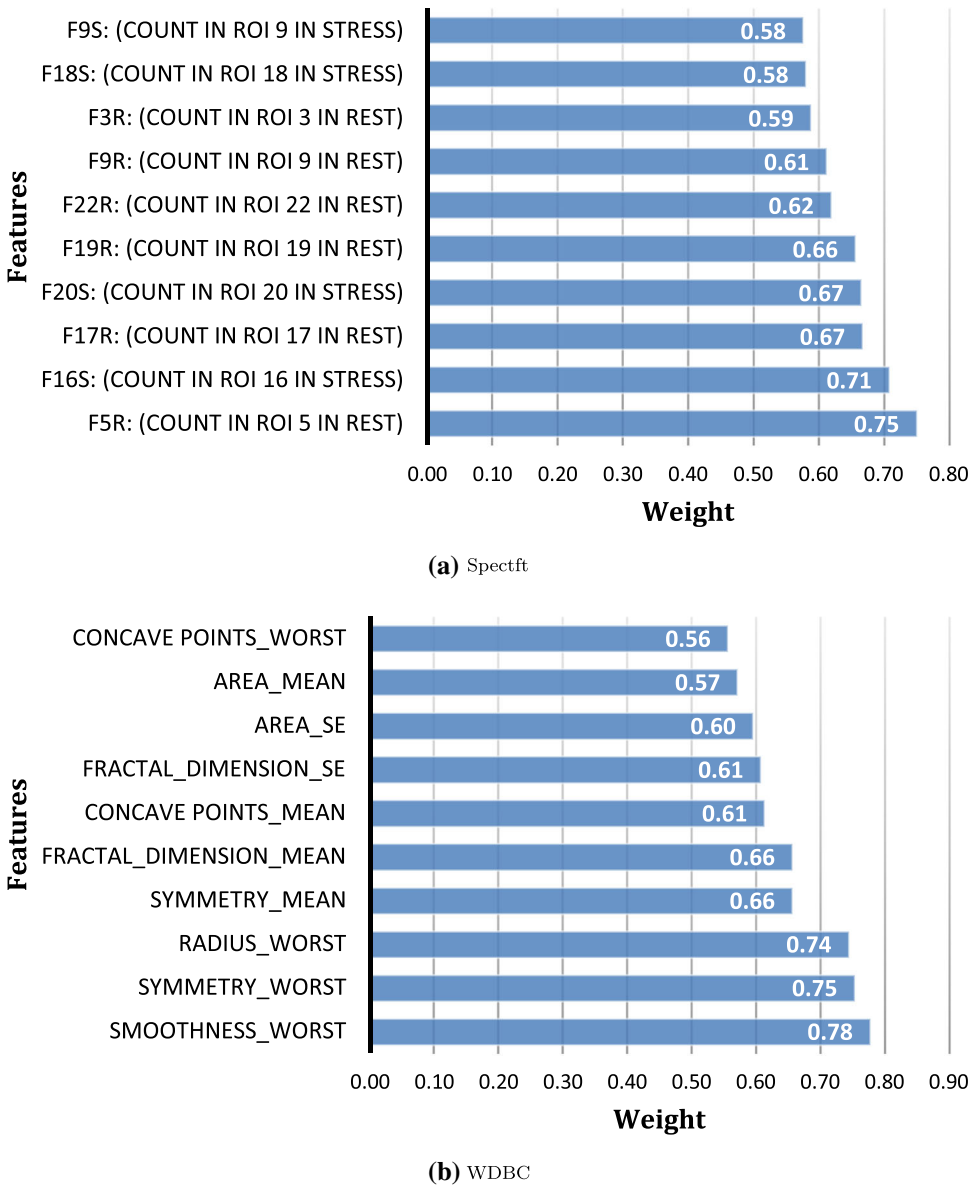


Fig. 13 Features weights obtained by CCSO method for Spectf and WDBC datasets

Table 9 Comparison of CCSO-SVM with standard classification models

Dataset	CCSO-SVM	J48	k-NN	NB	MLP	Grid-SVM
Blood	**78.4721**	76.2032	73.6631	73.3957	77.1390	76.5964
Colon cancer	**85.4762**	82.2581	77.4194	53.2258	72.5806	64.5238
Diabetes	**76.9498**	73.8281	70.1823	76.3021	75.3906	76.2867
Haberman	**74.8602**	73.5294	73.2026	74.8366	65.6863	72.2473
Heart	**85.5556**	80.9524	76.8707	83.6735	85.0340	81.4815
Libras	**89.7222**	69.7222	62.7778	62.7778	79.4444	78.6111
Liver	71.3277	68.6957	62.8986	55.3623	**71.5942**	68.1345
Parkinsons	95.9211	80.5128	**96.4103**	69.2308	90.7692	86.1579
Spectf	82.3789	85.3868	83.3811	70.7736	**90.5444**	79.4302
WDBC	**97.8916**	62.7417	71.7047	43.4095	71.7047	97.3653

Bold indicate the best results

Table 10 Comparison of CCSO-SVM with other methods in the literature

Data	CCSO-SVM	CSO-ELM	CSO-RELM	TMGWO2	GCACO	UPFS
Blood	78.4721	**0.7967**	0.7833	–	–	–
Colon Cancer	**85.4762**	–	–	–	0.8142	
Diabetes	76.9498	**0.7715**	0.7247	–	–	–
Haberman	**74.8602**	0.7302	0.7295	–	–	–
Heart	85.5556	**0.8612**	0.8402	0.8407	–	0.8389
Libras	89.7222	–	–	–	–	–
Liver	71.3277	**0.7322**	0.6910	–	–	–
Parkinsons	**95.9211**	0.9060	0.8891	0.8684	–	0.9023
Spectf	**82.3789**	0.7824	0.7861	0.7615	–	–
WDBC	**97.8916**	0.9613	0.9502	0.9482	0.9414	0.9641

– The results of this dataset are not available
Bold indicate the best results

optimization process when the algorithm is applied on large datasets.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Abdel-Basset M, El-Shahat D, El-henawy I, de Albuquerque VHC, Mirjalili S (2020) A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection. Expert Syst Appl 139:112824

Al-Zoubi A, Alqatawna J, Faris H, Hassonah MA (2019) Spam profiles detection on social networks using computational intelligence methods: the effect of the lingual context. J Inf Sci. https://doi.org/10.1177/0165551519861599

Ala'M AZ, Faris H, Alqatawna J, Hassonah MA (2018) Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts. Knowl Based Syst 153:91–104

Ala'M AZ, Heidari AA, Habib M, Faris H, Aljarah I, Hassonah MA (2020) Salp chain-based optimization of support vector machines and feature weighting for medical diagnostic information systems. In: Mirjalili S, Faris H, Aljarah I (eds) Evolutionary machine learning techniques. Springer, Berlin, pp 11–34

Aljarah I, Ala'M AZ, Faris H, Hassonah MA, Mirjalili S, Saadeh H (2018) Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cognit Comput 10(3):478–495

Bao Y, Hu Z, Xiong T (2013) A pso and pattern search based memetic algorithm for svms parameters optimization. Neurocomputing 117:98–106

Boardman M, Trappenberg T (2006) A heuristic for free parameter optimization with support vector machines. In: International joint conference on neural networks, 2006. IJCNN'06. IEEE, pp 610–617

Bouraoui A, Jamoussi S, BenAyed Y (2018) A multi-objective genetic algorithm for simultaneous model and feature selection for support vector machines. Artif Intell Rev 50(2):261–281

Chen WN, Zhang J, Lin Y, Chen N, Zhan ZH, Chung HSH, Li Y, Shi YH (2013) Particle swarm optimization with an aging leader and challengers. IEEE Trans Evol Comput 17(2):241–258

Cheng R, Jin Y (2015) A competitive swarm optimizer for large scale optimization. IEEE Trans Cybern 45(2):191–204

Dadaneh BZ, Markid HY, Zakerolhosseini A (2016) Unsupervised probabilistic feature selection using ant colony optimization. Expert Syst Appl 53:27–42

Eshtay M, Faris H, Obeid N (2018) Improving extreme learning machine by competitive swarm optimization and its application for medical diagnosis problems. Expert Syst Appl 104:134–152

Faris H, Hassonah MA, Ala'M AZ, Mirjalili S, Aljarah I (2018) A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture. Neural Comput Appl 30(8):2355–2369

Friedrichs F, Igel C (2005) Evolutionary tuning of multiple svm parameters. Neurocomputing 64:107–117

Guo X, Yang J, Wu C, Wang C, Liang Y (2008) A novel ls-svms hyper-parameter selection based on particle swarm optimization. Neurocomputing 71(16–18):3211–3215

Hsu CW, Lin CJ (2002) A simple decomposition method for support vector machines. Mach Learn 46(1):291–314

Hsu CW, Chang CC, Lin CJ (2003) A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, University of National Taiwan, Taipei, pp 1–12

Huang CL (2009) Aco-based hybrid classification system with feature subset selection and model parameters optimization. Neurocomputing 73(1–3):438–448

Huang CL, Dun JF (2008) A distributed pso-svm hybrid system with feature selection and parameter optimization. Appl Soft Comput 8(4):1381–1391

Huang CL, Wang CJ (2006) A ga-based feature selection and parameters optimization for support vector machines. Expert Syst Appl 31(2):231–240

LaValle SM, Branicky MS, Lindemann SR (2004) On the relationship between classical grid search and probabilistic roadmaps. Int J Robot Res 23(7–8):673–692

Li C, An X, Li R (2015) A chaos embedded gsa-svm hybrid system for classification. Neural Comput Appl 26(3):713–721

Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml

Lin SW, Ying KC, Chen SC, Lee ZJ (2008) Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Syst Appl 35(4):1817–1824

Liu Y, Wang G, Chen H, Dong H, Zhu X, Wang S (2011) An improved particle swarm optimization for feature selection. J Bionic Eng 8(2):191–200

Lorena AC, De Carvalho AC (2008) Evolutionary tuning of svm parameter values in multiclass problems. Neurocomputing 71(16–18):3326–3334

Moradi P, Rostami M (2015) Integration of graph clustering with ant colony optimization for feature selection. Knowl Based Syst 84:144–161

Phan AV, Le Nguyen M, Bui LT (2017) Feature weighting and svm parameters optimization based on genetic algorithms for classification problems. Appl Intell 46(2):455–469

Reif M, Shafait F, Dengel A (2012) Meta-learning for evolutionary parameter optimization of classifiers. Mach Learn 87(3):357–380

Sadiq AS, Faris H, Ala'M AZ, Mirjalili S, Ghafoor KZ (2019) Fraud detection model based on multi-verse features extraction approach for smart city applications. In: Rawat DB, Ghafoor KZ (eds) Smart cities cybersecurity and privacy. Elsevier, Berlin, pp 241–251

Shen L, Chen H, Yu Z, Kang W, Zhang B, Li H, Yang B, Liu D (2016) Evolving support vector machines using fruit fly optimization for medical data classification. Knowl Based Syst 96:61–75. https://doi.org/10.1016/j.knosys.2016.01.002

Shin KS, Lee TS, Hj K (2005) An application of support vector machines in bankruptcy prediction model. Expert Syst Appl 28(1):127–135

Sun G, Rong X, Zhang A, Huang H, Rong J, Zhang X (2019) Multi-scale mahalanobis kernel-based support vector machine for classification of high-resolution remote sensing images. Cognit Comput. https://doi.org/10.1007/s12559-019-09631-5

Tahir MA, Bouridane A, Kurugollu F (2007) Simultaneous feature selection and feature weighting using hybrid tabu search/k-nearest neighbor classifier. Pattern Recogn Lett 28(4):438–446

Tanveer M (2015) Robust and sparse linear programming twin support vector machines. Cognit Comput 7(1):137–149

Tharwat A, Hassanien AE (2018) Chaotic antlion algorithm for parameter optimization of support vector machine. Appl Intell 48(3):670–686

Tharwat A, Hassanien AE (2019) Optimizing support vector machine parameters using bat optimization algorithm. In: Hassanien AE (ed) Machine learning paradigms: theory and application. Springer, Berlin, pp 351–374

Tu CJ, Chuang LY, Chang JY, Yang CH et al (2007) Feature selection using pso-svm. Int J Comput Sci 33(1):1–3

Vapnik VN (1999) An overview of statistical learning theory. IEEE Trans Neural Networks 10(5):988–999

Wang M, Chen H (2020) Chaotic multiswarm whale optimizer boosted support vector machine for medical diagnosis. Appl Soft Comput 88:105946. https://doi.org/10.1016/j.asoc.2019.105946

Wu Q, Wu S, Liu J (2010) Hybrid model based on svm with gaussian loss function and adaptive Gaussian pso. Eng Appl Artif Intell 23(4):487–494

Xiaofang Y, Yaonan W (2008) Parameter selection of support vector machine for function approximation based on chaos optimization. J Syst Eng Electron 19(1):191–197

Xu Y, Guo R, Wang L (2013) A twin multi-class classification support vector machine. Cognit Comput 5(4):580–588

Yang Y, Pedersen JO (1997) A comparative study on feature selection in text categorization. Icml 97:412–420

Yuan SF, Chu FL (2007) Fault diagnostics based on particle swarm optimisation and support vector machines. Mech Syst Signal Process 21(4):1787–1798

Zhang X, Chen X, He Z (2010) An aco-based algorithm for parameter optimization of support vector machines. Expert Syst Appl 37(9):6618–6628

Zhang X, Fan M, Wang D, Zhou P, Tao D (2020) Top-k feature selection framework using robust 0-1 integer programming. IEEE Trans Neural Networks Learn Syst. https://doi.org/10.1109/TNNLS.2020.3009209

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

Zhao X, Li D, Yang B, Ma C, Zhu Y, Chen H (2014) Feature selection based on improved ant colony optimization for online detection of foreign fiber in cotton. Appl Soft Comput 24:585–596