Neural Computing and Applications (2020) 32:6925–6938
https://doi.org/10.1007/s00521-019-04159-z

ORIGINAL ARTICLE

Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm

Alaa Tharwat[1] · Thomas Gabel[1]

Received: 29 March 2018 / Accepted: 19 March 2019 / Published online: 1 April 2019

Abstract
The parameters of support vector machines (SVMs) such as kernel parameters and the penalty parameter have a great influence on the accuracy and complexity of the classification models. In the past, different evolutionary optimization algorithms were employed for optimizing SVMs; in this paper, we propose a social ski-driver (SSD) optimization algorithm which is inspired from different evolutionary optimization algorithms for optimizing the parameters of SVMs, with the aim of improving the classification performance. To cope with the problem of imbalanced data which is one of the challenging problems for building robust classification models, the proposed algorithm (SSD-SVM) was enhanced to deal with imbalanced data. In this study, eight standard imbalanced datasets were used for testing our proposed algorithm. For verification, the results of the SSD-SVM algorithm are compared with grid search, which is a conventional method of searching parameter values, and particle swarm optimization (PSO). The experimental results show that the SSD-SVM algorithm is capable of finding near-optimal values of SVMs parameters. The results also demonstrated high classification performance compared to the PSO algorithm.

Keywords Optimization algorithms · Support vector machine (SVM) · Parameter optimization · Imbalanced data

1 Introduction

Support vector machines (SVMs) are among the well-known machine learning techniques that are used for classification and regression problems [1, 2]. SVM classifier has been used in different applications such as biometrics [3], renewable energy [4], and cheminformatics [5]. In SVMs, the training data are utilized to build a classification model. Next, the classification model is used to classify an unknown sample. SVMs have a penalty parameter which determines the trade-off between minimizing the training error rate and maximizing the classification margin [6]. Moreover, SVMs have kernel parameters which define the nonlinear transformation from the input feature space to a higher-dimensional space. Hence, the kernel parameters have an influence on the performance of SVM and choosing suitable values for the two types of parameters, i.e., penalty and kernel parameters, controls the performance of SVM [2].

Evolutionary optimization algorithms have achieved competitive results when solving optimization problems including parameter tuning problem [6–8]. In this paper, an evolutionary optimization technique called social ski-driver (SSD) is proposed. The name of this method tributes to the fact that its stochastic nature bears some similarity to alpine skiing paths. Besides, the behavior of the SSD algorithm is also inspired by different evolutionary optimization algorithms such as the particle swarm optimization (PSO) algorithm [9], the gray wolf optimization (GWO) [10], and sine cosine algorithm (SCA) [11]. After all, the proposed SSD algorithm has the same stochastic nature as other swarm intelligence algorithms; hence, it has no guarantees for finding an optimal solution for any problem; however, it will often find a good solution if one exists.

✉ Alaa Tharwat
aothman@fb2.fra-uas.de

Thomas Gabel
tgabel@fb2.fra-uas.de

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

The problem of imbalanced data is one of the challenging problems to build a robust classification model. In imbalanced data, the number of samples of one class, i.e., the majority class, is outnumbering the samples of the other classes, i.e., the minority classes. Traditional classification algorithms tend to classify the unknown samples to the majority class; and hence, such a classification model is ineffective at classifying minority class samples, which is frequently the class of interest. In this paper, the synthetic minority over-sampling technique (SMOTE) algorithm was employed to obtain balanced data [12].

The main objectives of this paper are to:

- introduce the proposed SSD algorithm,
- use the SSD algorithm to present a novel SSD-SVM model for parameters tuning of SVMs classifier, which (1) improves the classification performance and (2) makes the optimal separating hyperplane obtainable in different classification problems, and
- employ the SSD-SVM algorithm for the classification of imbalanced data.

The rest of the paper is organized as follows: Sect. 2 introduces related work. In Sect. 3, the background of support vector machines, the SSD algorithm, and the SMOTE algorithm are presented. The proposed model (SSD-SVM) is introduced in Sect. 4. Experimental results and discussion are presented in Sect. 5. Concluding remarks and future work are provided in Sect. 6.

2 Related Work

There are many studies which searched for SVM parameters empirically by trying a finite number of values and keeping the values that obtain the best results. However, this procedure requires an exhaustive search over the whole search space to find feasible solutions [13]. A grid search was used to search for optimizing SVM parameters over the parameter space, where the parameters vary with a fixed step size through the parameter space [14]. The grid search algorithm is only suitable for optimizing a few parameters, since it is complex and time-consuming [2, 13, 15]. Different studies used evolutionary optimization algorithms for finding the optimal values of SVM parameters to achieve high classification performance. For example, Subasi employed the PSO algorithm to search for SVM parameters for diagnosis of neuromuscular disorders in EMG signals [16]. Also, the PSO algorithm was utilized for parameter determination of the SVM classifier and for feature selection [6]. Wu et al. proposed a hybrid genetic algorithm (GA) for optimizing SVM parameters for predicting the maximum electrical daily load [17]. The ant colony optimization (ACO) algorithm has also been used for optimizing SVM parameters [7]. In another research, Tharwat et al. used bat algorithm (BA) for optimizing SVMs [18]. Recently, the whale optimization algorithm (WOA) was employed for optimizing SVM classifier, where the combined WOA-SVM algorithm was used for the classification of toxicity effects of biotransformed hepatic drugs [5]. Dragonfly algorithm (DFA) was also used for optimizing SVM, and the proposed model was used to predict the toxicity effects of a new drug [19]. Further, Aydin et al. used a multi-objective artificial immune algorithm for optimizing SVM parameters [20]. Bat algorithm, firefly algorithm, fruit fly optimization algorithm, PSO, univariate marginal distribution algorithm (UMDA), and Boltzmann-UMDA were employed for finding the optimal SVM parameters in [21]. Tharwat et al. introduced the chaotic ant lion optimizer (CALO) for optimizing SVM parameters [22].

3 Preliminaries

3.1 Support Vector Machines (SVMs) Classifier

Support vector machines (SVMs) are among the most widely used learning algorithms. The main idea of them is to separate different classes using hyperplanes. However, the performance of SVMs is much affected by the non-linearly separable data and this problem can be addressed using kernel functions. The goal of kernel functions is to map the current features onto a higher-dimensional space where the data can be linearly separable. Selecting the appropriate kernel function and adjusting its parameters are two challenges of using SVMs. Computationally, finding the best decision plane is considered an optimization problem to help the kernel functions for finding the optimal space where the classes can be linearly separated through a nonlinear transformation [15].

3.1.1 Separable data (non-overlapping classes)

Assume we have N training samples ($X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$), where $\mathbf{x_i}$ indicates the ith training sample with d features and it is in one of two classes $y_i \in \{\pm 1\}$; hence, the training data are $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, where y_i is the class label for $\mathbf{x}_i$. The decision boundary between the two classes is denoted by a hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ when the data are linearly separable, where $\mathbf{w}$ is a weight vector, $\|\mathbf{w}\|$ represents the Euclidean norm of $\mathbf{w}$, b indicates the bias or threshold, and $\mathbf{x}$ is the input vector or the training sample. The feature space is divided by the hyperplane into two spaces, namely the *positive* half space, where the samples from the positive class, ω_+, are located, and the

negative half space, where the samples from the negative class, ω_-, are located [2].

In SVMs, the values of $\mathbf{w}$ and b are calculated to orientate the hyperplane to (1) be as far as possible from the closest samples, and (2) 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 the positive class and $\mathbf{w}^T\mathbf{x}_i + b \leq -1$ for the negative class. The equations of these two planes can be written as follows:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \quad \forall i = 1, 2, \ldots, N \tag{1}$$

The two planes are at the same distance from the hyperplane, and the distance between the two planes represents the margin of SVMs, $d_1 + d_2 = \frac{2}{\|\mathbf{w}\|}$, where d_1 and d_2 are the distances from H_1 and H_2, respectively, to the hyperplane. The margin of SVMs needs to be maximized subject to Eq. (1) as follows:

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

Equation (2) represents a quadratic programming problem that is formalized into Lagrange formula by combining the objective function ($\min \frac{1}{2}\|\mathbf{w}\|^2$) and the constraints ($y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0$) as follows:

$$\begin{aligned} \min L_P &= \frac{\|\mathbf{w}\|^2}{2} - \sum_i \alpha_i (y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) \\ &= \frac{\|\mathbf{w}\|^2}{2} - \sum_i \alpha_i y_i (\mathbf{w}^T\mathbf{x}_i + b) + \sum_{i=1}^{N} \alpha_i \end{aligned} \tag{3}$$

where $\alpha_i \geq 0$ indicates the Lagrange multiplier for the training sample $\mathbf{x}_i$ and L_P represents the primal problem [23]. The dual problem of SVMs can be formulated as follows:

$$\begin{aligned} &\max L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ &\text{s.t. } \alpha_i \geq 0, \ \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \forall i = 1, 2, \ldots, N \end{aligned} \tag{4}$$

where L_D is the dual form of L_P which needs to be maximized instead of minimizing L_P. As indicated in Eqs. (3 and 4), in the primal problem of SVMs, the objective function is minimized with respect to $\mathbf{w}$ and b; on the other hand, the objective function of the dual SVMs problem is maximized with respect to α_i [2].

In SVMs, the training samples with nonzero α's represent support vectors (SVs), which are the samples that achieve the maximum width margin, i.e., the closest to the separating hyperplane.

3.1.2 Non-separable data (overlapping classes)

The classification performance will be decreased when the data are non-separable. Thus, the constraints of linear SVMs must be relaxed and this can be achieved by adding a slack variable, ϵ_i, as follows:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \epsilon_i \geq 0 \quad \text{where} \quad \epsilon_i \geq 0 \tag{5}$$

where ϵ_i is the distance between the training sample (x_i) and the corresponding margin hyperplane; hence, it should be minimized as follows:

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

where C is the penalty or regularization parameter and it controls the trade-off between the size of the margin and the slack variable penalty. Equation (6) is formalized into Lagrange formula as follows:

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

where $\alpha_i \geq 0$.

3.1.3 Nonlinear separable data

In nonlinearly separable data, the kernel functions were employed to map the data from the current feature space into a higher-dimensional space using a nonlinear function, ϕ. The kernel function is defined as a dot product of the nonlinear functions as follows, $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$; and the objective function of SVMs will be:

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \epsilon_i \\ &\text{s.t. } y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) - 1 + \epsilon_i \geq 0 \quad \forall i = 1, 2, \ldots, N \end{aligned} \tag{8}$$

There are different kernel functions such as (1) linear kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, (2) *radial basis function* (RBF), $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$, and (3) polynomial kernel of degree d, $K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$ [24].

The focus of this study is to optimize the SVM classifier with the RBF kernel function. RBF kernel has only a single parameter σ. This parameter affects the mapping

transformation of the data space and hence changing the value of σ controls the performance of SVMs.

3.1.4 Illustrative example

The aim of this example is to show how the σ and C parameters influence the performance of SVMs. Given two classes, each class consists of 150 samples and hence the total number of samples is 300; 80% (240 samples) were used to train the SVMs model and the rest of the samples, i.e., 20% (60 samples), were utilized for testing the trained model.

This example has two experiments, Fig. 1 shows the results of the first experiment when the value of C was 10 and the values of σ were 0.01, 0.1, and 1, and Table 1 lists the results of this experiment. From the results, we can conclude that:

- In terms of a number of support vectors, all the training samples were used as SVs when σ was 0.01, and the number of SVs decreased when the value of σ increases. This reflects how the small value of σ increases the complexity of the SVM classifier and the model becomes much more sensitive to the noise in the training data.
- In terms of training accuracy, decreasing the value of σ makes the model more complex, i.e., less bias and high variance; this is called overfitting. Table 1 shows that the training error increases by increasing σ. Figure 1a shows also how the model was complex when σ was small and increasing σ relaxes the model as shown in Fig. 1b, c. Hence, very large σ may lead to the underfitting problem. Mathematically, the RBF kernel function is defined as follows: $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$; thus, small values of σ decrease the value of the kernel function. On the other hand, increasing σ increases the kernel value regardless the value of $-||\mathbf{x}_i - \mathbf{x}_j||^2$ and hence all samples become closer and this leads to the underfitting problem. As a consequence of that, with a large σ the training samples are overlapped because it limits the VC dimension. On the contrary, a small σ increases the VC dimension and hence the samples are perfectly separated [25].
- In terms of testing accuracy, as mentioned in the previous two points, small values of σ, e.g., in our example $\sigma = 0.01$, may lead to overfitting and hence to a small testing accuracy. On the other hand, increasing the value of σ makes the model more flexible, which yields an increase in the testing accuracy.

Figure 2 shows the results of our second experiment when the value of σ was one and the values of C were 0.01, 1, and 100, and Table 2 lists the results of this experiment. From the results, we can conclude that:

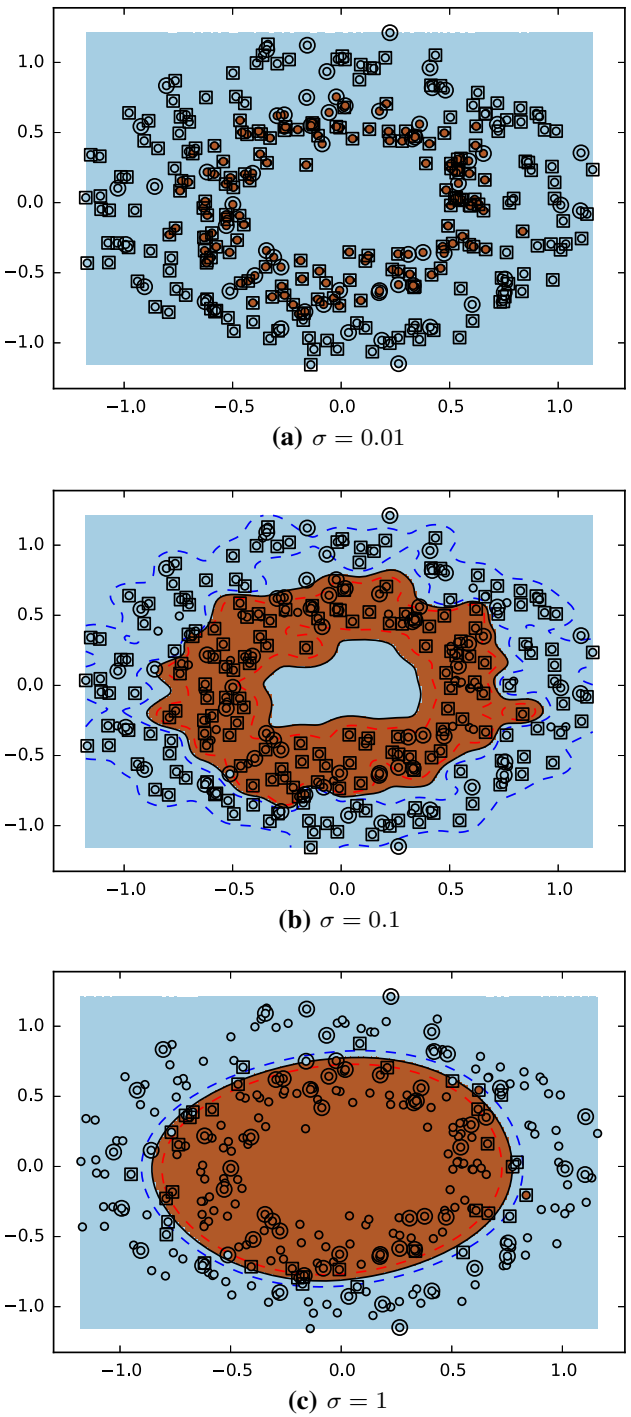


Fig. 1 The effect of σ parameter on the performance of SVMs with RBF kernel function (our first example). The x and y axes represent the first and second features, respectively. Decision boundaries (solid black line), two planes (dashed lines, red for the first class and blue for the second class), support vectors samples are marked by surrounding it by square shapes, and the testing samples are marked by surrounding it by circles (color figure online)

Table 1 The number of SVs (# SVs), training accuracy (Tr. Acc.), and the testing accuracy (Test Acc.) of the SVMs classifier with RBF kernel using different values of σ and $C = 10$

Results	$\sigma = 0.01$	$\sigma = 0.1$	$\sigma = 1$
# SVs	240	192	36
Tr. Acc.	$(240/240) = 100\%$	$(240/240) = 100\%$	$(233/240) \approx 97.08$
Test Acc.	$(32/60) \approx 53.33\%$	$(58/60) \approx 96.67\%$	$(59/60) \approx 98.33\%$

- In terms of a number of support vectors, 213 training samples were used as SVs when $C = 0.01$ and the number of SVs decreased when the value of C increases. Figure 2 shows that the margin width was large with a small C and increasing C reduces the margin width and this agrees mathematically with Eq. (7). This is the reason why small C increases the number of support vectors. This means that a small value of C leads to underfitting.
- In terms of training accuracy, small values of C result in minimum training accuracy. These results are in agreement with the first point, small C values increase the bias which may lead to severe underfitting.
- In terms of testing accuracy, the minimum results were obtained when $C = 0.01$, and the classification accuracy improved when the value of C was increased.

To sum up, the values of the σ and C parameters play an important role in controlling the flexibility of the resulting SVMs in fitting the data. Stated differently, these two parameters may lead to overfitting or underfitting.

3.2 Social ski-driver (SSD) optimization algorithm

In what follows, we propose a novel optimization algorithm, which is called social ski-driver (SSD) algorithm. The behavior of SSD was inspired from many different evolutionary optimization algorithms. Its name tributes to the fact that its stochastic exploration somehow resembles the paths that ski-drivers take downhill. SSD has many parameters; a brief description of these parameters is given below.

– *Positions of the agents* ($\boldsymbol{X}_i \in \mathcal{R}^n$): The positions of the agents are used to calculate the objective function at that location, where n is the dimension of the search space.
– *Previous best position* $\boldsymbol{P}_i$: The fitness value for all agents is calculated using the fitness function. The fitness value for each agent is then compared with its current position, and the best position is stored. This is similar to the PSO algorithm [9].
– *Mean global solution* $\boldsymbol{M}_i$: In our algorithm, as in the gray wolf optimizer (GWO) [10], the agents move toward the global point which represents the mean of the best three solutions (Fig. 3) as follows:

$$\mathbf{M}_i^t = \frac{X_\alpha + X_\beta + X_\gamma}{3}, \tag{9}$$

where X_α, X_β, and X_γ are the best three solutions.
– *Velocity of the agents* ($\mathbf{V}_i$): The agents' positions are updated by adding the velocity $\mathbf{V}_i$ as follows:

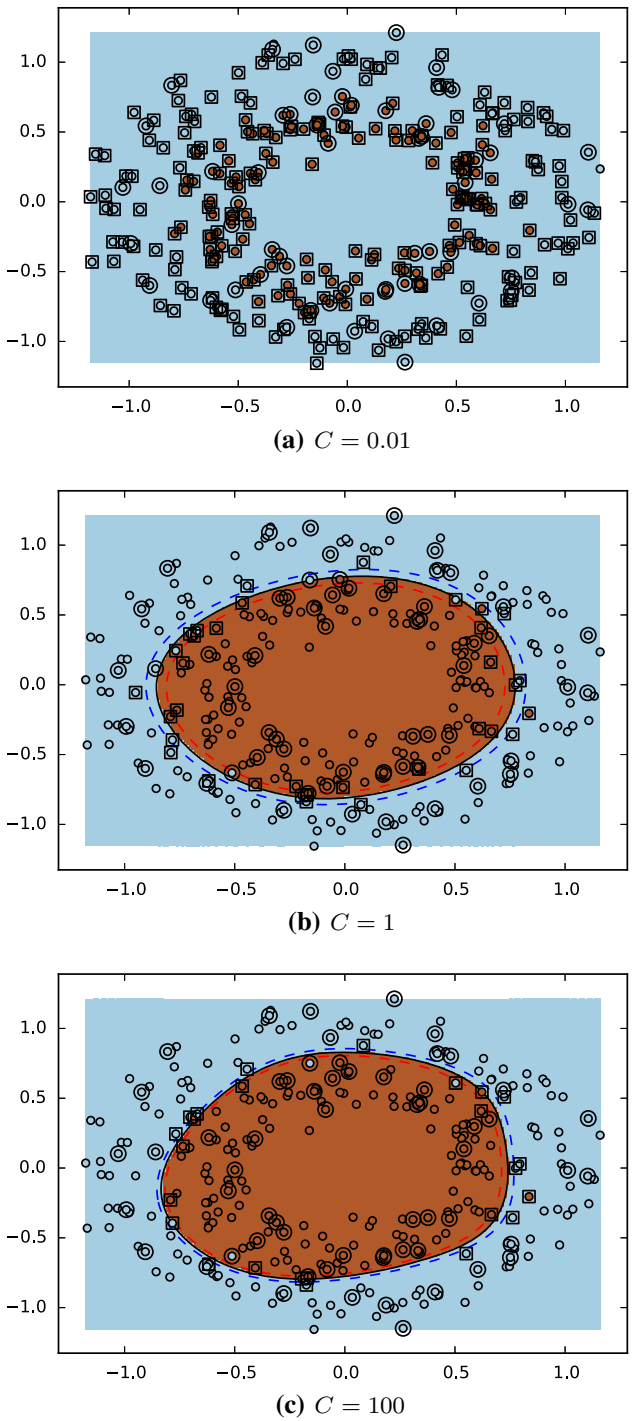
$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \mathbf{V}_i^t, \tag{10}$$

where

$$\mathbf{V}_i^{t+1} = \begin{cases} c\ \sin(r_1)(\mathbf{P}_i^t - \mathbf{X}_i^t) + \sin(r_1)(\mathbf{M}_i^t - \mathbf{X}_i^t) & \text{if } r_2 \le 0.5 \\ c\ \cos(r_1)(\mathbf{P}_i^t - \mathbf{X}_i^t) + \cos(r_1)(\mathbf{M}_i^t - \mathbf{X}_i^t) & \text{if } r_2 > 0.5 \end{cases} \tag{11}$$

where V_i is the velocity of X_i, r_1 and r_2 are uniformly generated random numbers in the range of [0, 1], $\mathbf{P}_i$ is the best solution of the ith agent, M_i is the mean global solution for the whole population, and c is a parameter which is used to make a balance between exploration and exploitation and it is calculated as follows: $c^{t+1} = \alpha c^t$, where t is the current iteration and $0 < \alpha < 1$ is used to reduce the value of c. Hence, $c \to 0$, where $t \to t_{\max}$ and $t_{\max}$ is the maximum number of iterations. As indicated in Eq. (11), the moving directions for the agents are not straightforward as in GWO or PSO, and this is because of the sine and cosine functions. Figure 3 visualizes a simple example of how two agents are moved in the SSD algorithm. This gives the proposed algorithm a better-guided exploration ability and makes the search directions to be diversified, but in a guided mode.

The main objective of the SSD is to search in the space for optimal or near-optimal solutions. The number of parameters that are needed to be optimized determines the dimension of that space. In SSD, the positions ($\mathbf{X}_i$) of agents are randomly initialized, where the number of agents is determined by the user. The agents update their positions by adding a velocity to their old positions as in Eq. (10). The agents' velocities are also randomly initialized, and it is modified according to Eq. (11). As indicated



(a) $C = 0.01$

(b) $C = 1$

(c) $C = 100$

Fig. 2 The effect of C parameter on the performance of SVMs with RBF kernel function ($\sigma = 1$) (our second example). The x and y axes represent the first and second features, respectively. Decision boundaries (solid black line), two planes (dashed lines, red for the first class and blue for the second class), support vectors samples are marked by surrounding it by square shapes, and the testing samples are marked by surrounding it by circles (color figure online)

in Eq. (11), the adjusted velocity of the agents depends on (1) the distance between the current position, $\mathbf{X}_i^t$, and the previous best position $\mathbf{P}_i$, (2) the distance between the current position, $\mathbf{X}_i^t$, and the mean global solution $\mathbf{M}_i$. Hence, the agents in SSD move toward the mean of the best three solutions which makes the SSD algorithm more social than PSO. Additionally, the agents in SSD are moved not in a straightforward direction which gives the SSD algorithm better exploration capabilities. The steps of SSD are listed in Algorithm (1).

Algorithm 1 SSD algorithm

1: Initialize the agents' positions X_i and velocities V_i. Assume the fitness function is minimum.
2: **while** stopping criteria are not met **do**
3: **for all** agents **do**
4: Calculate the fitness values.
5: Sort the agents according to their fitness values.
6: Calculate previous best position and mean global solution.
7: Generate a new solution by updating the agents' positions as denoted in Eq. (10).
8: Adjust the velocities of the agents as in Eq. (11).
9: **end for**
10: **end while**
11: Return the best solution.

3.3 Synthetic minority over-sampling technique (SMOTE) algorithm

The problem of imbalanced datasets appears when the number of samples of one class [this is called the *majority* class (S_{maj})] is significantly higher than the samples of the other class [this is called the *minority* class (S_{min})] [12]. This problem decreases the classification performance because it is difficult for a learning model to learn from a minority class and hence the minority class samples are misclassified frequently.

There are many methods which handle the imbalanced data problem such as kernel-based methods [12], cost-sensitive methods [26], and sampling methods [12]. This paper uses the sampling methods to obtain more balanced samples in each class. There are many well-known sampling methods such as *random under-sampling* (RUS), *random over-sampling* (ROS), and *synthetic minority over-sampling technique* (SMOTE). The goal of the RUS method is to extract randomly a small set of the majority class samples and remove the rest of majority samples to obtain balanced data. This reduces the training data. However, the removed samples may have useful information and this is the reason why RUS may decrease the classification performance [12]. The aim of the ROS method is to replicate the minority class samples. However, making exact copies of minority class samples may increase the classification performance but without extending the decision boundary of the minority class [12, 27].

Table 2 The number of SVs (# SVs), training accuracy (Tr. Acc.), and the testing accuracy (Test Acc.) of the SVMs classifier with different values of C and with the RBF kernel $\sigma = 1$

Results	$C = 0.01$	$C = 1$	$C = 100$
# SVs	213	86	20
# Tr. Acc.	$(122/240) \approx 50.83\%$	$(233/240) \approx 97.08\%$	$(235/240) \approx 97.92$
# Test Acc.	$(28/60) \approx 46.67\%$	$(58/60) \approx 96.67\%$	$(58/60) \approx 96.67\%$

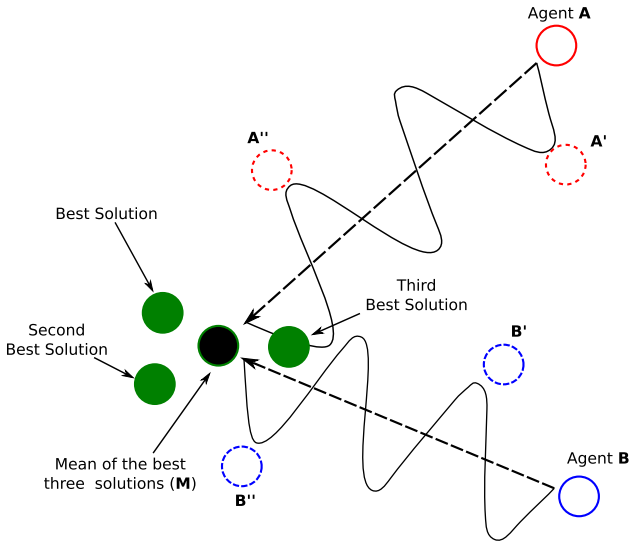


Fig. 3 Illustration of an example of how two agents (A and B) using the SSD algorithm move toward the mean of the best three solutions (**M**). The agent **A** moves toward **M** in a nonlinear direction to $\mathbf{A}'$ and then to $\mathbf{A}''$, and similarly, the agent **B**

In SMOTE algorithm, the goal is to generate new minority class samples based on the similarities between current minority samples. In the minority class $S_{\min}$, for each sample $x_i \in S_{\min}$, the k nearest samples are selected, and a new synthetic sample is created according to $x_{\text{new}} = x_i + r_{ij} \times \delta = x_i + (\hat{x}_{n(nk)} - x_i) \times \delta$, where x_i indicates one of the minority class samples, $\hat{x}_{n(nk)}$ is one of the k nearest neighbors for $x_i : \hat{x}_{n(nk)} \in S_{\min}$, nk represents a random number between 1 and k to select one of the k nearest neighbors randomly, k is the number of selected neighbors, $\delta \in [0, 1]$ represents a random number, and x_{new} is the new sample along the line joining x_i and $\hat{x}_{n(nk)}$. Hence, a synthetic or new sample is generated randomly by selecting one of the k nearest neighbors, $\hat{x}_{n(nk)}$, and by multiplying a random number, δ, with the corresponding feature vector difference, r_{ij}, and then adding this vector to x_i [28]. SMOTE is better than other sampling algorithms because (1) it extends the decision region of the minority class and (2) it preserves all data without removing any samples [28].

4 The Proposed model: SSD-SVM

As shown in Fig. 4, the first step in our approach is the data preprocessing. The goal of this step is to adopt linear scaling to avoid features in greater numeric ranges dominating those in smaller numeric ranges. This step helps to get higher classification performance [18]. In our model, the *Min-Max* method is used to scale the feature ranges to [0, 1] according to $f' = \frac{f - \min}{\max - \min}$, where f is the original value, min and max are the lower and upper bounds of the feature value, respectively, and f' is the scaled value. Next, the SMOTE algorithm is used to obtain balanced data as shown in Fig. 4. The next steps of the proposed algorithm are presented in the next sections.

4.1 Parameters' initialization

In this step, the parameters of SSD such as the number of agents and the maximum number of iterations are initialized. In the proposed model, the SSD provides the SVM classifier with the parameters' values, i.e., C and σ, to train the SVM classifier using the training data. In our model, there are two parameters; therefore, the search space is two-dimensional and each point in the space represents a combination between C and σ. The agents' positions are

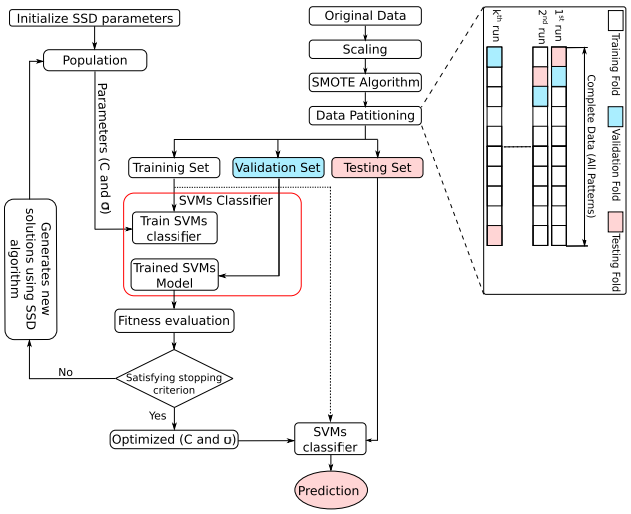


Fig. 4 Flowchart of the proposed model (SSD-SVM)

initialized randomly, and the searching range of parameter C is bounded by $C_{\min} = 0.01$ and $C_{\max} = 1000$, and the searching range of σ is bounded by $\sigma_{\min} = 0.01$ and $\sigma_{\max} = 50$. Increasing these limits extends the search space; thus, more agents are required for searching for the optimal solution, which results in more computational time and a slower convergence rate.

4.2 Fitness evaluation

The models that were proposed in [7, 16, 18] divided the data into training and testing sets only; and they used the training set for training the model and the testing set for evaluating the model. This may lead to the overfitting problem. By contrast, in our model, the data were partitioned into three sets. The training set is used for the training, and the validation set is used to evaluate the trained model during the iterations; finally, the testing set is used to test the final model. In other words, the training and validation sets were used for building a classification model and determining suitable parameters for it, whereas the fully trained SVM model is then tested using the testing set. Hence, our model will be evaluated using totally unseen data.

Having trained a machine learning model on imbalanced data, the accuracy or misclassification rate may lead to erroneous conclusions because the accuracy does not distinguish between the numbers of corrected labels of different classes. Therefore, in the proposed model, due to the small number of samples in the minority class[1] the aim is to maximize the sensitivity, S, as follows:

$$\text{Minimize} : \mathcal{F} = -S = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{12}$$

where TP is the number of true positives and TN is the number of true negatives. Hence, for each agent's position, the training set is used to train the SVM classifier, while the validation set is used to calculate the sensitivity rate. The optimal solution is found in a position where the values of C and σ achieve the maximum sensitivity. When the termination conditions are satisfied, the algorithm ends; otherwise, we proceed with the next generation operation. The proposed algorithm is terminated when the maximum number of iterations is reached.

5 Experimental Results and Discussion

In this section, we present the results of different experiments that were conducted to evaluate the performance of the proposed SSD-SVM algorithm. The platform adopted to test the SSD-SVM algorithm is a PC with the following features: Intel(R) Core (TM) i5-2400 CPU3.10GHz, 4G RAM, a Windows 7 operating system, and MATLAB 7.10.0 (R2010a). To evaluate the proposed algorithm, eight classification datasets were used. The datasets were obtained from the KEEL[2] collection of datasets, and they were divided into two divisions. The first division contains the datasets which have an *imbalance ratio* (IR) lower than nine (called *Lower IR*), whereas the second division contains the datasets which have an IR higher than nine (is called *Upper IR*). The IR represents the number of instances of the majority class for each instance of the minority class. Moreover, all datasets have two classes. The descriptions of all datasets are listed in Table 3, where the top four datasets represent the lower IR datasets and the bottom four datasets are the upper IR datasets.

In all experiments, k-fold cross-validation tests have been used. In k-fold cross-validation, the data were randomly divided into k subsets of (approximately) equal size and the experiment is run k times. For each run, one subset was used as a testing set, and one is used as a validation set, and the other $k - 2$ sets were used as a training set (Fig. 4). The average of the k results from the folds can then be calculated to produce a single estimation. In our experiments, tenfold cross-validation was used to estimate the results of each approach, and the obtained results are illustrated in the form of *average* $\pm$ *standard deviation*. Moreover, in all experiments, the number of iterations of SSD was 50 and the number of agents was 20.

Three assessment methods were used in our experiments, namely sensitivity (Sen.) (Sect. 4.2), specificity (Spc.), and the area under curve (AUC). The specificity is the percentage of the correctly classified negative samples. It is defined as $\text{TNR} = \frac{\text{TN}}{\text{TN}+\text{FP}}$, where TN is the number of true negatives and FP represents the number of false positives. The AUC indicates the area under the receiver operating characteristics (ROC) curve, and it is calculated as $\text{AUC} = \frac{1+\text{TPR}-\text{FPR}}{2}$ [29].

5.1 Experimental results

In this section, the results of two experiments are presented. The goal of the first experiment (Sect. 5.1.1) is to compare the SSD-SVM with a naive grid search over the SVM parameters. In the second experiment (Sect. 5.1.2), the aim is to compare the SSD-SVM with the PSO-SVM [6] and BA-SVM algorithm [18].

[1] In our model, the minority class is considered as the positive class.

[2] Available at http://sci2s.ugr.es/keel/imbalanced.php.

Table 3 Datasets characterization

Dataset	# Samples	# Features	IR
Ecoli2 (D1)	336	7	5.46
Ecoli3 (D2)	336	7	8.6
Glass0 (D3)	214	9	2.06
Glass6 (D4)	214	9	6.38
Glass4 (D5)	214	9	15.46
Glass2 (D6)	214	9	11.59
winequality-white-9_vs_4 (D7)	168	11	32.6
Poker Hand (D8)	244	10	29.5

5.1.1 SSD-SVM vs. Grid-SVM

The aim of this experiment is to compare the proposed algorithm with naive grid search. Grid search as mentioned before has been used for adjusting or tuning SVM parameters [6, 18, 30].

Table 4 shows the sensitivity, specificity, and AUC results of this experiment. As can be read from that table, the sensitivity of the proposed algorithm is much higher than the results achieved using grid search. For further comparison, the nonparametric Wilcoxon signed-rank test was used for all datasets. As shown in Table 4, in terms of sensitivity, the p value for D4 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. In terms of specificity and AUC results, our proposed algorithm obtains results better than the grid search-optimized SVMs; and the p values for all datasets are smaller than the significant level of 0.005. Generally, compared with basic grid search, the proposed SSD-SVM achieves high sensitivity, specificity, and AUC results. It is also worth mentioning that the required computational time for the proposed SSD-SVM algorithm is much less than the time required for the grid search algorithm. This is because the computational cost for the grid search algorithm increases exponentially with (1) the number of parameters, (2) the range, and (3) the number of sampling points for each parameter [31].

5.1.2 SSD-SVM versus PSO-SVM and BA-SVM

The second experiment was conducted to compare our proposed SSD-SVM algorithm with PSO-SVM that was proposed in [6] and BA-SVM that was introduced in [18]. Tables 5 and 6 show the results of this experiment.

As shown, the SSD-SVM algorithm yields higher sensitivity results than the PSO-SVM and BA-SVM algorithms in most cases. Moreover, using the PSO-SVM algorithm, the p values for D1 and D5 are larger than the predicted statistical significance level of 0.005, but the other p values are smaller than the significance level of 0.005. In terms of specificity, the SSD-SVM algorithm obtains results better than the PSO-SVM algorithm; and only the p values for D4 and D5 are larger than the predicted statistical significance level of 0.005. The SSD-SVM algorithm also achieves better AUC results than the PSO-SVM algorithm; and the p values for D1 and D5 are larger than the predicted statistical significance level of 0.005. Table 6 shows that with the BA-SVM algorithm, in terms of sensitivity results, the p values for D1, D3, and D5 are larger than the predicted statistical significance level of 0.005, but the other p values are smaller than the significance level of 0.005. Additionally, in terms of specificity and AUC results, the SSD-SVM algorithm obtains results better than the BA-SVM algorithm, and only the p values for D1 and D5 are larger than the predicted statistical significance level of 0.005.

For further comparison between the SSD-SVM and the PSO-SVM and BA-SVM algorithms, Fig. 5 shows the average of results for all datasets. As shown, the SSD-SVM outperforms the PSO-SVM and BA-SVM algorithms. Generally, the SSD-SVM obtains results better than the PSO-SVM and BA-SVM algorithms. This is due to the following reasons:

1. The particles in the PSO algorithm move toward the global best position or previous best positions and in the bat algorithm, the particles/agents move toward the global best position, while in SSD the agents move toward the mean of the best three solutions. This is beneficial for the SSD algorithm in terms of being more social than PSO and BA. Hence, if the global best solution is trapped into a local minimum, the other two best solutions can help the SSD to escape from the local minimum solution.
2. The agents in the PSO and BA algorithms move in straightforward directions, while in SSD, the moving directions are not straightforward due to the sine and cosine functions as shown in Fig. 3 yielding ski-driver-live paths through the search space. This gives the SSD algorithm better exploration capabilities.
3. The PSO algorithm has fixed parameters which need to be tuned first, while the parameters of BA or SSD can be changed as the iterations proceed. Hence, SSD and BA switch automatically from exploration to exploitation. As a results, both algorithms (i.e., BA and SSD) have the ability to escape from local minimum solutions better than the PSO algorithm.

As a consequence, it is not surprising that SSD behaves more efficiently than PSO and BA algorithms.

In terms of computational time, a comparison of the SSD-SVM, PSO-SVM, and BA-SVM reveals that all

Table 4 Comparison between the SSD-SVM and grid search SVMs algorithms in terms of sensitivity, specificity, and AUC

Dataset	IR	Grid Search SVMs			SSD-SVM			p value for Wilcoxon testing		
		Sen.	Spc.	AUC	Sen.	Spc.	AUC	Sen.	Spc.	AUC
D1	5.46	0.90 ± 0.08	0.93 ± 0.03	0.92 ± 0.04	0.97 ± 0.05	0.94 ± 0.02	0.95 ± 0.04	< 0.005	< 0.005	< 0.005
D2	8.6	0.88 ± 0.08	0.91 ± 0.03	0.89 ± 0.05	0.92 ± 0.08	0.94 ± 0.04	0.91 ± 0.06	< 0.005	< 0.005	< 0.005
D3	2.06	0.85 ± 0.10	0.90 ± 0.02	0.88 ± 0.05	0.92 ± 0.04	0.91 ± 0.02	0.92 ± 0.03	< 0.005	< 0.005	< 0.005
D4	6.38	0.76 ± 0.05	0.81 ± 0.04	0.80 ± 0.04	0.78 ± 0.12	0.82 ± 0.08	0.81 ± 0.10	0.0054	< 0.005	< 0.005
D5	15.46	0.89 ± 0.13	0.97 ± 0.02	0.95 ± 0.06	0.94 ± 0.02	1 ± 0.00	0.96 ± 0.02	< 0.005	< 0.005	< 0.005
D6	11.59	0.96 ± 0.02	0.97 ± 0.02	0.96 ± 0.02	0.98 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	< 0.005	< 0.005	< 0.005
D7	32.6	0.93 ± 0.04	0.97 ± 0.02	0.95 ± 0.03	0.97 ± 0.03	1 ± 0.00	0.98 ± 0.01	< 0.005	< 0.005	< 0.005
D8	29.5	0.84 ± 0.05	0.88 ± 0.06	0.85 ± 0.04	0.86 ± 0.07	0.92 ± 0.04	0.87 ± 0.06	< 0.005	< 0.005	< 0.005

Table 5 Comparison between the SSD-SVM and PSO-SVM algorithms in terms of sensitivity, specificity, and AUC

Dataset	PSO-SVM			SSD-SVM			p value for Wilcoxon testing		
	Sen.	Spc.	AUC	Sen.	Spc.	AUC	Sen.	Spc.	AUC
D1	0.97 ± 0.03	0.93 ± 0.02	0.95 ± 0.03	0.97 ± 0.05	0.94 ± 0.02	0.95 ± 0.04	0.0064	< 0.005	0.0051
D2	0.91 ± 0.04	0.88 ± 0.06	0.89 ± 0.05	0.92 ± 0.08	0.94 ± 0.04	0.91 ± 0.06	< 0.005	< 0.005	< 0.005
D3	0.90 ± 0.03	0.88 ± 0.02	0.90 ± 0.03	0.92 ± 0.04	0.91 ± 0.02	0.92 ± 0.03	< 0.005	< 0.005	< 0.005
D4	0.73 ± 0.08	0.79 ± 0.06	0.78 ± 0.06	0.78 ± 0.12	0.82 ± 0.08	0.81 ± 0.10	< 0.005	0.006	< 0.005
D5	0.94 ± 0.03	1 ± 0.00	0.97 ± 0.01	0.94 ± 0.02	1 ± 0.00	0.96 ± 0.02	0.0062	0.0059	0.0058
D6	0.96 ± 0.02	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	< 0.005	< 0.005	< 0.005
D7	0.95 ± 0.03	1 ± 0.00	0.97 ± 0.02	0.97 ± 0.03	1 ± 0.00	0.98 ± 0.01	< 0.005	< 0.005	< 0.005
D8	0.84 ± 0.06	0.86 ± 0.04	0.85 ± 0.04	0.86 ± 0.07	0.92 ± 0.04	0.87 ± 0.06	< 0.005	< 0.005	< 0.005

Table 6 Comparison between the SSD-SVM and BA-SVM algorithms in terms of sensitivity, specificity, and AUC

Dataset	BA-SVM			SSD-SVM			p value for Wilcoxon testing		
	Sen.	Spc.	AUC	Sen.	Spc.	AUC	Sen.	Spc.	AUC
D1	0.98 ± 0.02	0.94 ± 0.03	0.96 ± 0.03	0.97 ± 0.05	0.94 ± 0.02	0.95 ± 0.04	0.0063	0.0058	0.0064
D2	0.91 ± 0.03	0.89 ± 0.05	0.90 ± 0.04	0.92 ± 0.08	0.94 ± 0.04	0.91 ± 0.06	< 0.005	< 0.005	< 0.005
D3	0.92 ± 0.02	0.90 ± 0.03	0.91 ± 0.03	0.92 ± 0.04	0.91 ± 0.02	0.92 ± 0.03	0.0060	< 0.005	< 0.005
D4	0.72 ± 0.05	0.78 ± 0.07	0.76 ± 0.06	0.78 ± 0.12	0.82 ± 0.08	0.81 ± 0.10	< 0.005	< 0.005	< 0.005
D5	0.94 ± 0.04	1 ± 0.00	0.96 ± 0.03	0.94 ± 0.02	1 ± 0.00	0.96 ± 0.02	0.0061	0.0060	0.0059
D6	0.97 ± 0.04	0.98 ± 0.02	0.98 ± 0.03	0.98 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	< 0.005	< 0.005	< 0.005
D7	0.94 ± 0.02	0.99 ± 0.01	0.97 ± 0.04	0.97 ± 0.03	1 ± 0.00	0.98 ± 0.01	< 0.005	< 0.005	< 0.005
D8	0.85 ± 0.05	0.88 ± 0.04	0.86 ± 0.05	0.86 ± 0.07	0.92 ± 0.04	0.87 ± 0.06	< 0.005	< 0.005	< 0.005

algorithms require approximately the same computational time.

It is also worth mentioning that the proposed algorithm obtained high classification results with all datasets. Furthermore, the obtained results for the datasets which have high IR were better than the results achieved for datasets with low IR. This reflects the robustness of our proposed algorithm against imbalanced data. Another important finding is that our algorithm improved the sensitivity results without sacrificing the specificity results. It is interesting to note from our experiments that the optimal value of σ changes from one dataset to another. This is because the transformation of the samples to a higher-dimensional space depends also on the distance between samples. As a consequence, the range of features has a great influence on the value of σ.

[3] http://archive.ics.uci.edu/ml/index.php.

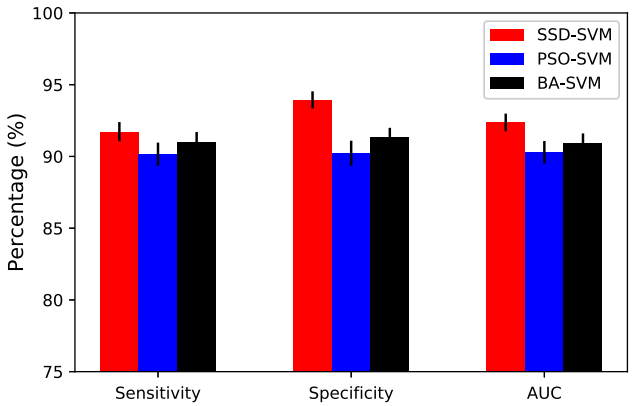


Fig. 5 Comparison between the proposed SSD-SVM algorithm and the PSO-SVM and BA-SVM algorithms

Finally, different experiments have been conducted to investigate the performance of our proposed algorithm against:

- *Balanced data:* In this experiment, two well-known balanced datasets (iris and liver from the UCI[3] data repository) were used to test the performance of our proposed algorithm when applied for balanced data. The liver dataset has two classes, and only two classes were used from the iris dataset. Compared with the results in [32], the SSD-SVM algorithm obtained competitive results as listed in Table 7. These results prove that our model is also applicable for balanced data.
- *Data with high dimensionality:* As shown in Table 3, the dimensionality of all datasets that were used in our experiments was low (<11). A simple experiment was conducted to test our proposed algorithm for one dataset (sonar dataset from the UCI repository) with higher dimensionality. Table 7 shows empirically that our proposed algorithm achieves promising results compared to the results in [6]. However, with imbalanced data which have high IR, more samples from the minority class are generated which increases the required computational time. One of the solutions for this problem is to assign different weights to the imbalanced classes.
- *Multiclass data:* In our experiments, all the datasets have only two classes. As an additional test, our proposed algorithm was applied on a multiclass dataset (iono dataset from the UCI repository). The iono dataset (as shown in Table 7) has three classes with different numbers of samples. In order to be applicable for multiclass datasets, we developed an extended SSD-SVM version. To this end, the main modification was made in the SMOTE part of the algorithm. As mentioned before (Sect. 3.3), in the binary classification problem, the SMOTE algorithm is used for generating new minority class samples to obtain more balanced data for both the majority and minority classes. In a multiclass problem, one majority class, which has the maximal number of samples, was selected, and then all minority class samples were oversampled using the SMOTE algorithm. The results of this experiment (shown in Table 7) obtained competitive results compared with the results in [32].

To sum up, the developed SSD-SVM algorithm yielded more appropriate SVMs parameters, giving better results across different datasets. Even so, we cannot guarantee that the SSD-SVM algorithm performs well and outperforms other methods in different applications. In fact, many factors may have an influence on the quality of the proposed model, such as the number of data samples, the diversity of training sets, representativeness, severe imbalance ratios, and the number of features.

6 Conclusions and future work

The SVMs learning algorithm is widely used in different applications. However, the parameter values of SVMs have a great influence on the classification performance of SVMs. This study proposes a social ski-driver-based approach (SSD-SVM) for optimizing SVM parameters. Because imbalanced datasets represent a severe challenge in many supervised learning scenarios, in this paper, the proposed algorithm was developed to be suitable for imbalanced data. In our algorithm, the minority class represents the positive class; hence, the goal is to maximize the sensitivity of the SVM model. Different experiments were conducted to compare the proposed SSD-SVM algorithm with a naive grid search for the parameters of the SVMs and PSO-SVM algorithms by applying many standard classification datasets with different imbalance ratios. The results of this study showed that the SSD-SVM

Table 7 Experimental results of the proposed algorithm with different datasets

Datasets	classes	#dim=4	Sen.	Spc.	AUC	Previous results
Iris	(50/50)	4	1 ± 0	1 ± 0	1 ± 0	Acc. (100%)
Liver	(145/200)	6	81.07 ± 0.90	71.18 ± 2.24	76.13 ± 1.84	Acc. (68.7%)
Sonar	(97/111)	60	86.47 ± 1.05	88.54 ± 1.18	87.23 ± 0.97	Acc. (88.3%)
Iono	(59/71/48)	13	92.12 ± 2.14	94.22 ± 0.84	93.73 ± 1.13	Acc. (97.92%)

algorithm outperformed the PSO-SVM as well as grid search algorithms in terms of sensitivity, specificity, and AUC.

There are several directions for future studies. Firstly, the experiments in this paper were performed using only eight datasets, but other public datasets should be tested in the future to verify and extend the proposed algorithm. Secondly, severe imbalance ratio datasets should be used to evaluate the proposed algorithm. Thirdly, due to the performance of SSD, it would be worthwhile to explore the potential of SSD to other applications.

Compliance with ethical standards

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

Appendix

Illustrative example

The goal of this example is to compare the proposed SSD algorithm with the PSO and bat algorithms. In this example, the number of agents was 20, the number of iterations was 50, and the search space was two-dimensional, i.e., $d = 2$. We used the Rastrigin objective function, which is defined as follows:

$$F(x_i) = 10d + \sum_{t=1}^{d}(x_i^2 - 10.\cos(2\pi x_i))), \quad -5.12 \leq x_i \leq 5.12 \tag{13}$$

The Rastrigin function is one of the well-known functions, and it has been used in different studies because it has many local optimal solutions and one global minimum (Fig. 6). The optimal solution is zero, and it is located at the origin as shown in Fig. 6 [10, 33]. In this experiment, the SSD, PSO, and bat algorithms were employed to find the optimal solution for the Rastrigin function, and the results of this experiment are summarized in Fig. 7. As shown, all algorithms were run for three runs, where the convergence curves of the SSD, PSO, and bat algorithms are represented in the red, blue, and green colors, respectively. For a fair comparison, in all runs, all algorithms started from the same position, i.e., the initial solutions for all optimization algorithms were the same. It is interesting to note from Fig. 7 that the SSD algorithm significantly escaped from local minima in many cases. This is clear in the figure when the convergence curve of SSD was constant for some iterations. As shown, after some iterations, the SSD algorithm escaped from this local minimum solution and converged to a better solution.

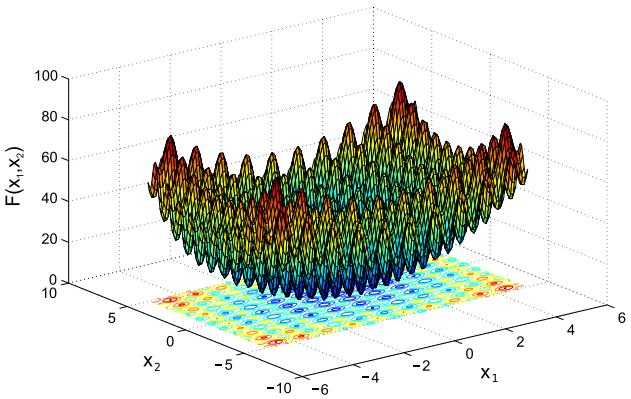


Fig. 6 The surface plot of the Rastrigin function

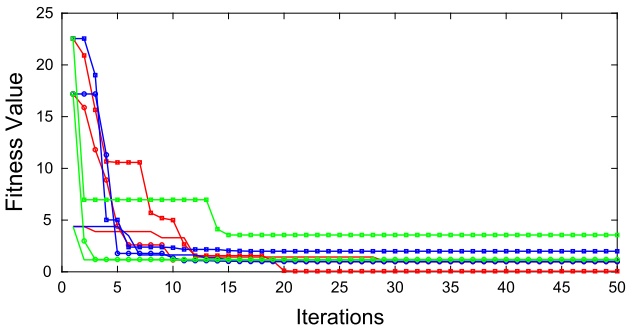


Fig. 7 Convergence curves of the SSD (in the red color), the PSO (in the blue color), and the bat (in the green color) algorithms in different three runs (color figure online)

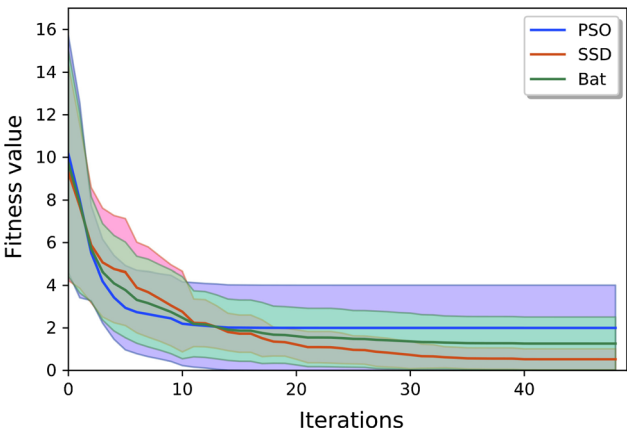


Fig. 8 A comparison between the SSD, bat, and PSO optimization algorithms using 20 runs. The solid lines represent the mean of all runs, and the shaded areas represent the standard deviation

On the other hand, the blue lines have no fluctuations because in two runs, the PSO algorithm succeeded to reach the optimal solution without facing local minimum solutions and in one run, the PSO algorithm trapped into one local minimum from the fifth iteration and it did not escape from it till the end of iterations. The bat algorithm is more flexible than PSO, and hence it (the bat algorithm) can escape from the local minimum better than PSO. This is because PSO has

fixed parameters that need to be tuned first, while the parameters of the bat algorithm can be adjusted as the iterations proceed.

Figure 8 shows a comparison between the SSD, PSO, and bat optimization algorithms using 20 runs. As shown, the average of runs of the SSD algorithm (i.e., the red solid line) was much lower than the PSO and bat algorithms. Further, the shaded area of the SSD algorithm which represents the standard deviation of all runs was much smaller than the shaded area of the other two algorithms. This reflects that the SSD algorithm is more stable than PSO and bat algorithms and most of the runs of the SSD algorithm converged to the optimal or near-optimal solution. Additionally, the bat algorithm obtained results better than PSO. This is due to the fixed parameters of PSO, and hence the bat algorithm has a quick convergence rate compared with PSO.

These results reflect how the SSD algorithm can keep a larger diversity which is capable of finding better results than PSO and bat algorithms. This is because (1) SSD is more social than PSO and bat algorithms because the agents in SSD track the mean of the best three solutions, while in the bat and PSO algorithms, the agents follow the global best solution, (2) the agents of SSD move not in a straightforward direction, while in PSO and bat algorithms the agents move only in a straightforward direction, which gives the SSD algorithm better exploration capabilities. Even so, we still cannot guarantee that the SSD algorithm must perform well and outperform other optimization algorithms in different applications.

References

1. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
2. Wang L (2005) Support vector machines: theory and applications, vol 177. Springer, Berlin
3. Wang Y, Wang Y, Tan T (2004) Combining fingerprint and voiceprint biometrics for identity verification: an experimental comparison. In: Biometric authentication, pp 289–294
4. Bouzerdoum M, Mellit A, Pavan AM (2013) A hybrid model (SARIMA-SVM) for short-term power forecasting of a small-scale grid-connected photovoltaic plant. Sol Energy 98:226–235
5. Tharwat A, Moemen YS, Hassanien AE (2017) Classification of toxicity effects of biotransformed hepatic drugs using whale optimized support vector machines. J Biomed Inform 68:132–149
6. 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
7. 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
8. Yamany W, Tharwat A, Hassanin M F, Gaber T, Hassanien AE, Kim TH (2015) A new multi-layer perceptrons trainer based on ant lion optimization algorithm. In: Fourth international conference on information science and industrial applications (ISI). IEEE, pp 40–45
9. Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intell 1(1):33–57
10. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
11. Mirjalili S (2016) Sca: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
12. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
13. Friedrichs F, Igel C (2005) Evolutionary tuning of multiple SVM parameters. Neurocomputing 64:107–117
14. 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
15. Chapelle O, Vapnik V, Bousquet O, Mukherjee S (2002) Choosing multiple parameters for support vector machines. Mach Learn 46(1–3):131–159
16. Subasi A (2013) Classification of EMG signals using pso optimized SVM for diagnosis of neuromuscular disorders. Comput Biol Med 43(5):576–586
17. Wu CH, Tzeng GH, Lin RH (2009) A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. Expert Syst Appl 36(3):4725–4735
18. Tharwat A, Hassanien AE, Elnaghi BE (2017) A ba-based algorithm for parameter optimization of support vector machine. Pattern Recognit Lett 93:13–22
19. Tharwat A, Gabel T, Hassanien AE (2017) Parameter optimization of support vector machine using dragonfly algorithm. In: International conference on advanced intelligent systems and informatics. Springer, pp 309–319
20. Aydin I, Karakose M, Akin E (2011) A multi-objective artificial immune algorithm for parameter optimization in support vector machine. Appl Soft Comput 11(1):120–129
21. Rojas-Domínguez A, Padierna LC, Valadez JMC, Puga-Soberanes HJ, Fraire HJ (2018) Optimal hyper-parameter tuning of svm classifiers with application to medical diagnosis. IEEE Access 6:7164–7176
22. Tharwat A, Hassanien AE (2018) Chaotic antlion algorithm for parameter optimization of support vector machine. Appl Intell 48:670–686
23. Kecman V (2001) Learning and soft computing: support vector machines, neural networks, and fuzzy logic models. MIT Press, Cambridge
24. Tharwat A (2019) Parameter investigation of support vector machine classifier with kernel functions. Knowl Inf Syst 1–34. https://doi.org/10.1007/s10115-019-01335-4
25. Burges CJ (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2(2):121–167
26. Sun Y, Kamel MS, Wong AK, Wang Y (2007) Cost-sensitive boosting for classification of imbalanced data. Pattern Recognit 40(12):3358–3378
27. Sun Y, Wong AK, Kamel MS (2009) Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell 23(04):687–719
28. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
29. Tharwat A (2018) Classification assessment methods. Appl Comput Inform. https://doi.org/10.1016/j.aci.2018.08.003
30. Huang CL, Wang CJ (2006) A GA-based feature selection and parameters optimizationfor support vector machines. Expert Syst Appl 31(2):231–240
31. Moore G, Bergeron C, Bennett KP (2011) Model selection for primal SVM. Mach Learn 85(1–2):175
32. Zhang Y, Zhang P (2015) Machine training and parameter settings with social emotional optimization algorithm for support vector machine. Pattern Recognit Lett 54:36–42

33. Tharwat A, Gaber T, Hassanien AE, Elnaghi BE (2017) Particle swarm optimization: a tutorial. In: Handbook of research on machine learning innovations and trends. IGI Global, pp 614–635