Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **8**, Number **4**, November **2023**

Deep Sparse Least Squares Support Vector Machines Based on the Sparrow Search Algorithm

Songsong Xu*

School of Information Engineering
Chengdu Vocational & Technical College of Industrys
Chengdu 610218, China
askibm@outlook.com

Shanshan Liu*

School of technology
Asia Pacific University of Technology & Innovation
Kuala Lumpur 56000, Malaysia
TP068598@mail.apu.edu.my

*Corresponding author: Songsong Xu
Received May 5, 2023, revised July 26, 2023, accepted September 8, 2023.

Abstract. *In order to solve the problems of diversity loss and easy to fall into local optimization in the late iteration of sparrow search algorithm, an Improved Sparrow Search Algorithm(ISSA) with multi-strategy is proposed. The population variety is boosted and the capability of global search is enhanced by the introduction of the Circle chaotic mapping to initialize the population; the Gaussian difference variation is introduced to improve the ability of the population to jump out of the local optimum; and the golden sine algorithm is introduced to balance the global search and local exploitation ability. In order to verify the effectiveness of the improved algorithm, five benchmark functions were selected for simulation experiments and compared with the genetic algorithm, grey wolf algorithm, particle swarm algorithm and sparrow search algorithm. The simulation results show that the IISSA has better search accuracy and convergence speed. The ISSA is used to determine the optimal metric properties and the optimal parameters of the Deep Sparse Least Squares Support Vector Machine (DS-LS-SVM) for the problem of difficult parameter selection in the classification. Simulation results on publicly available software defect datasets show that the ISSA-DS-LS-SVM model is able to retain valid and useful metric attribute data than traditional research methods, resulting in better performance evaluation metrics in software defect prediction applications.*
Keywords: Sparrow search algorithm; Chaotic mapping; Golden sine; Gaussian difference variation; Least squares support vector machine

1. **Introduction.** Nowadays, information technology has become deeply involved in people's work and life. As a bridge between people and technology, the quality of the presentation and implementation of the whole system directly determines whether the product can meet the customer's requirements. The consequences of software quality problems can be very frightening.

Deficiencies existing in the software's own system are the main problem affecting the quality of the system [1,2,3]. From the system developer's point of view, to ensure the quality of the software system, it is necessary to avoid the deficiencies in the system from the revelation stage of development, and to correct the existing deficiencies in time. The

system developer needs to focus on this level of software testing to fundamentally guarantee the correctness and safety of the system development, thus providing the fullest guarantee for the overall elimination of software deficiencies. Specifically, the main purpose of software testing is to find the deficiencies in the system and to see if the existing functionality is in line with the previous requirements [4,5]. The length of testing for software accounts for almost half of the entire development time, in other words, the workload in this area is enormous.

Software defect prediction is the solution to this problem, as it allows the evaluation of deficiencies in the system with the help of previous data. Software defect prediction avoids the need to spend too much effort on software testing and avoids wasting human and material resources. We have mainly classified software defect prediction into three types: dynamic, conventional static, and machine learning prediction [6,7,8]. Learning algorithms, have been the focus of machine learning based defect prediction models. Due to the development of machine learning techniques, the integration of learning algorithms and defect prediction models has been promoted. Many versions of software defect prediction based on machine learning prediction have emerged, the most classic of which is the software defect prediction model based on support vector machines (SVM) [9,10].

Support Vector Machine (SVM) is an efficient machine learning algorithm [11,12], which has gained widespread use and interest in recent years. However, when the feature space is very large, even infinite dimensional, traditional SVM models face the problem of unbearable memory and time complexity. Deep learning models are able to handle high-dimensional data, but too many parameters can make the models easily fall into overfitting. To this end, this paper proposes the Depth Sparse Least Squares Support Vector Machine (DS-LS-SVM), which can be effectively used for learning and classification of large-scale, high-dimensional data.

1.1. **Related Work.** In addition to support vector machines, researchers have experimented with a variety of classifiers to implement defect prediction models.

Wang and Yao [13] implemented twenty-seven prediction models with the WEKA toolbox and found that RF was the best. Yang et al. [14] performed predictions on the NASA dataset with eight models including conventional machine learning methods and neural networks, while comparing the simulation results. Ultimately, the algorithm SVM was shown to be the most effective by comparing metrics such as accuracy, recall and F-value. A machine learning-based approach to software defect prediction, which uses the relationship between system code metrics and system deficiencies as test evidence, has shown good results for the support vector machine approach and is considered one of the best choices for relevant assessments because of its classification accuracy.

However, traditional SVMs have limitations in dealing with high-dimensional large-scale data. To address this problem, Sparse Least Squares Support Vector Machine (LS-SVM) was proposed [15,16], which can effectively reduce the dimensionality of data and improve the computational efficiency.

Ma et al. [17] presented a two-stage feature selection method for LS-SVM based on correlation analysis and recursive feature elimination. Their experiments on different datasets showed that the proposed method provides better generalization performance compared to other feature selection methods. Arya et al. [18] introduced an online LS-SVM algorithm that allows for continuous fine-tuning of the model using new data while preserving the original sparsity structure. Results demonstrated that the proposed approach achieves comparable performance to batch training methods with significantly reduced computation time. Fan et al. [19] proposed a novel feature representation method that combines deep learning and sparse coding with LS-SVM for classification tasks. They

achieved improved classification accuracy on benchmark datasets compared to traditional feature extraction techniques.

The DS-LS-SVM takes a bottom-up learning approach, combining the sparse nature of deep learning with the L2-parametric penalty of SVMs. Specifically, the DS-LS-SVM first maps the input feature vector to a low-dimensional subregion a relatively low-dimensional subregion by using an L1-parametrization with weights. The L2-parametrics is then used to control the number of support vectors. Compared to the LS-SVM model, DS-LS-SVM not only compresses the feature information efficiently, but also maintains excellent performance in complex situations.

1.2. **Motivation and contribution.** DS-LS-SVM achieved improved classification accuracy on benchmark datasets compared to traditional feature. However, no strict criteria have been established for the selection of DS-LS-SVM parameters, and much relies on empirical or manual testing.

The main innovations and contributions of this work include

(1) An Improved Sparrow Search Algorithm (ISSA) incorporating multiple strategies is proposed to address the problems that the sparrow search algorithm decreases in population diversity and tends to fall into local optimum in the late iteration. By introducing the Circle chaotic mapping to initialize the population, the population diversity is increased and the global search ability is improved; the Gaussian difference variation is introduced to improve the ability of the population to jump out of the local optimum; and the golden sine algorithm is introduced to balance the global search and local exploitation ability.

(2) To address the problem of difficult parameter selection for DS-LS-SVM in classification, ISSA is used to determine the optimal metric attributes and the optimal parameters for DS-LS-SVM. The effectiveness of the proposed ISSA-DS-LS-SVM model is verified in a software defect prediction application.

2. SSA theory.

2.1. **Basic principles.** SSA mainly simulates the foraging and anti-predatory behaviour of sparrows, which can be divided into several different roles according to their behaviour [20]. Individuals with high fitness values in the population are called discoverers, with high predation ability and high energy levels of their own. The remaining individuals are called followers, while a certain proportion of individual sparrows from the population are selected as scouts to scout the population for early warning and abandon food if danger is detected and move to a safe location.

In the population, most of the discoverers gather in the centre of the population and have a high predatory capacity. At the same time, most sparrows in the population will move towards the centre of the high energy sparrows in order to obtain more food, known as followers. Individuals at the edge of the population are vulnerable to attack by other predators, so a certain number of sparrows are selected to act as scouts, calling to warn the population when a predator is spotted and constantly moving towards the centre of the population to renew their position to ensure safety.

The SSA algorithm is a new group intelligence algorithm inspired by sparrow foraging, anti-predation and vigilance behaviours [21,22], which consists of a discoverer, joiner and scout. The discoverer is responsible for foraging in the group, the joiner follows the best-adapted discoverer in order to obtain food, and the joiner also monitors the discoverer to wait for an opportunity to grab food. During foraging, if a scout senses a predator in danger, it will quickly alert the whole colony to fly to safety.

2.2. **Specific steps.** The execution flow of SSA is shown below:

(1) Initialize the location of the population.

The initial position of the population can be represented by a matrix with a total of n sparrows, each row of the matrix being the initial position of each sparrow in d-dimensional space:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,d} \end{bmatrix} \tag{1}$$

In d-dimensional space, the initial fitness values [23] for all its sparrows are:

$$\mathbf{f}_x = \begin{bmatrix} f([x_{1,1} \;\; x_{1,2} \;\; \dots \; x_{1,d}]) \\ f([x_{2,1} \;\; x_{2,2} \;\; \dots \; x_{2,d}]) \\ \vdots \\ f([x_{n,1} \;\; x_{n,2} \;\; \dots \; x_{n,d}]) \end{bmatrix} \tag{2}$$

(2) Update the finder location.

The sparrow with the highest fitness value in the population acts as a spotter and the spotter moves continuously to find more food, while other followers move with it. If a sparrow discovers a predator, it chirps to the population to indicate a warning, and if the warning signal value is greater than a threshold, the finder moves to a safe position, the followers move with it, and the whole population is updated to a new position, where the finder update formula can be expressed as

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t} \exp\left(\frac{-i}{\alpha \cdot iter_{\max}}\right), R_2 < S \\ X_{i,j}^{t} + Q \cdot \mathbf{L}, R_2 \geq S \end{cases} \tag{3}$$

where $X_{i,j}^{t+1}$ denotes the j-th dimensional position of the i-th individual in generation t of the population; α is a uniform random number in $(0, 1]$; $iter_{\max}$ is the maximum number of iterations; R_2 is the warning value at which the scout sparrow detects a predator calling; S is a predetermined safety value; Q is a standard normally distributed random number; and L is a $1 \times d$ dimensional unit matrix.

(3) Update follower position. Between 70% and 80% of the population is selected as discoverers [24], with all the rest being followers. Followers may also become discoverers as discoverers move towards more food and acquire more food. The follower position update equation can be expressed as:

$$X_{i,j}^{t+1} = \begin{cases} Q \cdot \exp\left(\frac{X_{\text{worst}}^{t} - X_{i,j}^{t}}{i^2}\right), i > n/2 \\ X_{p}^{t+1} + \left|X_{i,j}^{t} - X_{p}^{t+1}\right| \cdot \mathbf{A} \cdot \mathbf{L}, i \leq n/2 \end{cases} \tag{4}$$

where X_{worst}^{t} is the worst position of the sparrow in the current population; X_{p}^{t+1} is the best position in the population; A is a matrix where each element is randomly assigned a value of 1 or -1.

(4) Update scout locations.

10%-20% of the population is randomly selected to act as scouts, and while the other sparrows are foraging, they are responsible for keeping a lookout and will immediately abandon the current food to move to the next location if danger is approaching. The

formula for updating the scout's position can be expressed as:

$$X_{i,j}^{t+1} = \begin{cases} X_{\text{best}}^{t} + \beta \cdot \left| X_{i,j}^{t} - X_{\text{best}}^{t} \right|, f_i \neq f_g \\ X_{i,j}^{t} + K \cdot \left(\frac{\left| X_{i,j}^{t} - X_{\text{worst}}^{t} \right|}{(f_i - f_w) + \varepsilon} \right), f_i = f_g \end{cases} \tag{5}$$

where X_{best}^{t} is the current global optimal position; β is a random number satisfying a normal distribution; K is a random number with the value range $[-1, 1]$; $f_i$is the fitness value corresponding to the current optimal position; f_g is the fitness value corresponding to the current worst position; and ε is the minimum constant.

3. **Proposed ISSA.**

3.1. **Circle chaos mapping.**

Tent chaos mapping and Logistic chaos mapping are commonly used to initialize populations. In this paper, Circle chaos mapping is used to initialize the population.

The traditional logistic chaos mapping is not uniformly distributed and has an impact on the convergence speed and accuracy of the algorithm. Although the Tent mapping is more uniformly distributed, it has unstable cycles and tends to fall into immobility, while the Circle mapping is more stable and has a comparable uniformity of distribution to Tent. Circle chaotic mappings are a typical class of chaotic mappings and are one of the classical models in nonlinear dynamical systems.

$$X_{i+1} = \text{ mod } \left[X_i + 0.2 - \left(\frac{0.5}{2\pi} \right) \sin\left(2\pi X_i\right), 1 \right] \tag{6}$$

where X_{i+1} represents the $(i+1)$-th position. Mod[] is the residual function. Generate a successful initialization population for each position.

The Circle chaotic mapping is a continuous dynamical system described by a non-linear differential equation. The introduction of noise enhances the randomness and unpredictability of the Circle chaotic mapping, helping to improve encryption and immunity to interference. This can be achieved by adding Gaussian white noise:

$$x(n+1) = x(n) + (a - by(n)) \cdot dt + \sigma \cdot sqrt(dt) \cdot N(0,1) \tag{7}$$

$$y(n+1) = y(n) + (bx(n) - z(n) - xy(n)) \cdot dt + \sigma \cdot sqrt(dt) \cdot N(0,1) \tag{8}$$

$$z(n+1) = z(n) + (xy(n) - cz(n)) \cdot dt + \sigma \cdot sqrt(dt) \cdot N(0,1) \tag{9}$$

where $N(0,1)$ shows Gaussian white noise with mean 0 and variance 1; σ is the noise intensity parameter and dt is the time step. The above three equations describe the three-dimensional form of the Circle chaos mapping, with a, b and c being the mapping parameters respectively.

The addition of Gaussian white noise can simulate the interference and noise in the environment, effectively improving the resistance and unpredictability of the encryption. The size of the noise strength parameter σ needs to be determined according to the specific application scenario and needs to be adjusted experimentally to achieve the best results.

3.2. **Gaussian difference variation.**

To prevent the population from falling into a local optimum during iteration, the optimal position of the population is perturbed at each iteration using Gaussian difference variation.

Compared to traditional difference-in-variance algorithms, Gaussian difference is able to generate larger perturbations in the vicinity of the current variant individual, making it easier for the algorithm to jump out of the local optimum. If the position is better

after the perturbation, the population optimum position is updated, otherwise it remains unchanged. The Gaussian difference variance is calculated as follows:

$$L^t = k_1 \cdot g_1 \cdot \left(X^* - X^t_{\text{best}}\right) + k_2 \cdot g_2 \cdot \left(X_{\text{rand}} - X^t_{\text{best}}\right) \tag{10}$$

where k_1 and k_2 denote weight coefficients; g_1 and g_2 denote Gaussian distributed random numbers with mean 0 and variance 1; X^* is the population optimal sparrow position; X_{rand} denotes the random sparrow position; and denotes the position after Gaussian differential variation.

Once the perturbed positions are obtained, the fitness values of the positions before and after the perturbation are compared and the position with the best fitness is selected as the optimal position for this iteration of the population. The optimal position is updated as follows:

$$X^t_{\text{best}} = \begin{cases} L^t & , f\left(L^t\right) < f\left(X^t_{\text{best}}\right) \\ X^t_{\text{best}} & , f\left(L^t\right) \geq f\left(X^t_{\text{best}}\right) \end{cases} \tag{11}$$

where $f\left(L^t\right)$ and $f\left(X^t_{\text{best}}\right)$ represent the adaptation values of the post- and pre-disturbance positions respectively.

3.3. **The golden-sine algorithm.** The Golden Sine Algorithm (Golden-SA) [25] reduces the search space by the golden ratio to approximate the optimal solution of the algorithm, which has the advantage of simplicity of principle and superiority finding capability. Gold-SA algorithm position update formula is as follows:

$$X^{t+1}_{i,j} = X^t_{i,j} \cdot \left|\sin\left(r_1\right)\right| + r_2 \cdot \sin\left(r_1\right) \cdot \left|k_1 \cdot X^t_p - k_2 \cdot X^t_{i,j}\right| \tag{12}$$

The discoverer needs a larger search range in the early stage to improve the global search capability, while a smaller search range is needed in the later stage to mine the global optimal location, and the update of the follower location is also vulnerable to the update of the discoverer location. Therefore, the search capability of SSA is mainly related to the search range of the discoverer. However, the discoverer in SSA at $R_2 < S$, each dimension of the sparrow population is shrinking with the number of iterations, so the golden sine algorithm is used to improve Eq. (12), balancing the discoverer's ability to search globally in the early stages with the ability to exploit locally in the later stages.

$$X^{t+1}_{i,j} = \begin{cases} X^t_{i,j} \cdot \left|\sin\left(r_1\right)\right| + r_2 \cdot \sin\left(r_1\right) \cdot \left|k_1 \cdot X^t_p - k_2 \cdot X^t_{i,j}\right| & , R_2 < S \\ X^t_{i,j} + Q \cdot L & , R_2 \geq S \end{cases} \tag{13}$$

4. **ISSA-based DS-LS-SVM.**

4.1. **Fundamentals of LS-SVM.** SVM models typically exhibit slight generalization errors in the two test data sets when dealing with binary classification. Therefore, after establishing the linear discriminant equation, this equation can be used to test the type of each given test sample.

Suppose the linearly divisible sample set is $\{(x_i, y_i) | x_i \in R^m, y_i \in (-1, +1), i = 1, 2, \ldots, n\}$, then it can be expressed as a linear function as follows

$$g(x) = < \omega \cdot x > + b \tag{14}$$

The corresponding classification surface can then be expressed as follows:

$$< \omega \cdot x > + b = 0 \tag{15}$$

The optimization problem for the SVM model is calculated as follows:

$$\min\left(\frac{1}{2}\|\omega\|^2 + c\sum_{i=1}^{l}\varepsilon_i\right) \text{ s.t.} \quad y_i\left(\omega \cdot \emptyset\left(x_i\right) + b\right) \geq 1 - \varepsilon_i \tag{16}$$

The optimization problem for the LS-SVM model [26] is calculated as follows:

$$\min\left(\frac{1}{2}\|\omega\|^2+c\sum_{i=1}^{l}\varepsilon_i^2\right)\ \text{s.t.}\quad y_i\left(\omega\cdot\emptyset\left(x_i\right)+b\right)\geq 1-\varepsilon_i \tag{17}$$

where c is the penalty factor.

With the introduction of the kernel function, the optimal classification surface of the SVM is calculated as follows:

$$f(x)=\operatorname{sgn}\left(\sum_{i=1}^{n}a_i y_i K\left(x_i,x\right)+b\right) \tag{18}$$

In contrast to the above, the LS-SVM algorithm actually operates the following set of linear functions.

$$\begin{bmatrix} 0 & 1 & \ldots & y_1 \\ 1 & K\left(x_i,x_j\right)+1/c & \ldots & K\left(x_i,x_j\right) \\ \ldots & \ldots & \ldots & \ldots \\ 1 & K\left(x_i,x_j\right) & \ldots & K\left(x_i,x_j\right)+1/c \end{bmatrix}\times\begin{bmatrix} b \\ a_1 \\ a_1 \\ a_1 \end{bmatrix}=\begin{bmatrix} 0 \\ y_1 \\ \ldots \\ y_n \end{bmatrix} \tag{19}$$

Ultimately the optimal classification surface for the LSSVM model can be obtained.

$$\min\left\{\frac{1}{2}\sum_{i,j=1}^{n}K\left(x_i,x_j\right)+\sum_{i=1}^{n}\frac{a_i^2}{2\gamma}-\sum_{i=1}^{n}x_i y_i+b\sum_{i=1}^{n}a_i\right\} \tag{20}$$

Although the LS-SVM algorithm is able to obtain the corresponding classification results and achieve the task objectives when processing general classification problems, it still has two significant drawbacks [27,28]. 1. The solution process of LS-SVM is more similar to linear computation, which feels simple, but the reality is that it is computationally more complicated when processing large-scale data classification; 2, LS-SVM algorithm does not possess sparsity, which leads to the classification algorithm test after training, often showing lower rate and much less efficiency.

4.2. **DS-LS-SVM.** DS-LS-SVM is an effective solution to the above two drawbacks. Unlike the traditional sparse LS-SVM, DS-LS-SVM first sets up the hidden layer unit, and subsequently uses the data structure obtained from the shallow layer as the basis for deep continuous training learning, and continuously performs the above steps. This learning process focuses on the process of obtaining features, and its feature proposal structure is shown in Figure 1.

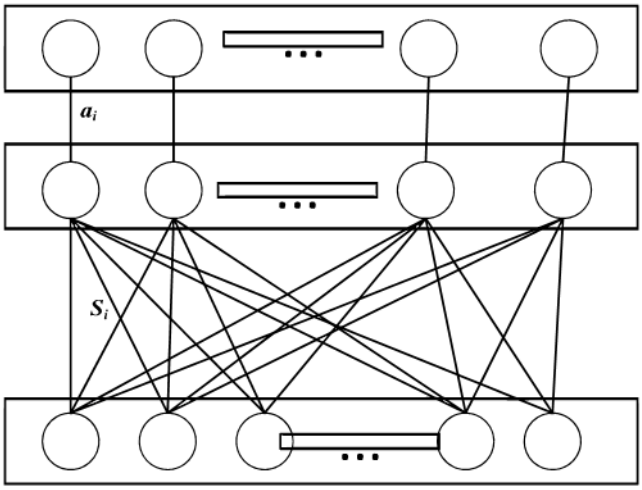

Figure 1. Feature extraction model of sparse LS-SVM

The highest layer learning result obtained is then the final learning result of the classifier. The model after such training is called a stacked sparse LS-SVM model, also known as DS-LS-SVM model, as shown in Figure 2. DS-LS-SVM contains a number of hidden layers, each of which obtains its function from its next level hidden layer. The training results of the lower level hidden layers are used to calculate the activation values to obtain the upper level feature values. Subsequently, mapping is used to obtain the corresponding high-dimensional data which are used as input features for the higher hidden layers. The inputs are executed sequentially and extracted to obtain the learning results of the DS-LS-SVM model. If the N M-dimensional data information is denoted as $x_1, \ldots, x_n$,

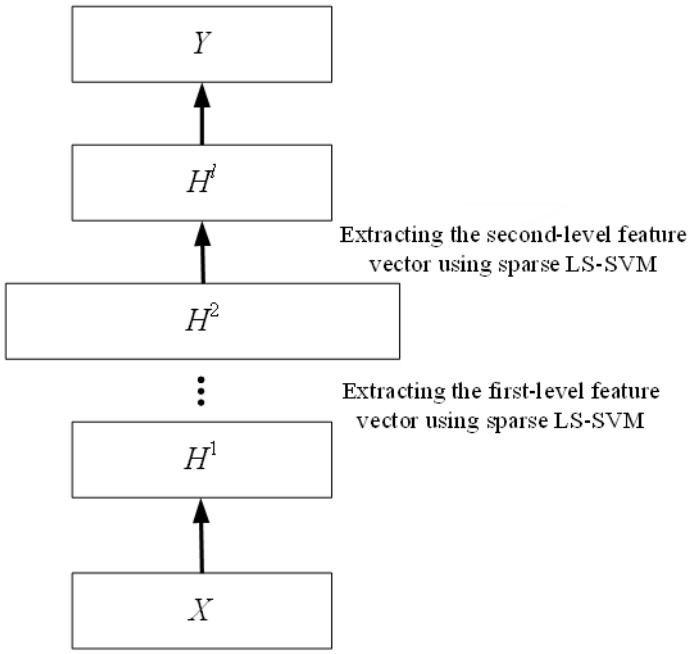

Figure 2. Model architecture of DS-LS-SVM

then the N training data can be obtained as P M-dimensional support vectors, denoted

by $s_1, \ldots, s_p$. The Lagrangian multiplier corresponding to each support vector is denoted by $a_1, \ldots, a_p$ and its category label is denoted by $t_1, \ldots, t_p$.

$$h^1(i) = a_i t_i K\left(s_i, x\right) \tag{21}$$

where $h^1(i)$ denotes the i-th element in the first hidden unit.

A scientifically valid detection of new data points is then required by virtue of the activation kernel function. This data point is passed to the next layer by means of a mapping and the sample type is analysed at the top layer in the form of a classification function.

$$y(x) = \sum_{i=1}^{l} a_i t_i K\left(s_i, \ddot{O}(x)\right) + b \tag{22}$$

where s_i represents the i-th support vector; l represents the number of support vectors at the end of the network; and $\ddot{O}$ is the feature vector of the test data x transformed with the help of the hidden layer.

Through the above analysis, DS-LS-SVM has a higher algorithmic superiority compared to LS-SVM, not only in terms of the optimisation problem solved, but also in terms of the sparsity of the solution.

4.3. **ISSA-DS-LS-SVM.** However, the DS-LS-SVM model suffers from the problem of difficult parameter selection, which also leads to inefficient search. In addition, and due to the problem of premature sieving of beneficial attribute information, DS-LS-SVM also suffers from the disadvantage of being prone to local optimal solutions. For this reason, this work proposes to use an improved ISSA to achieve parameter optimisation and metric element attribute selection optimisation for DS-LS-SVM.

The improved ISSA not only excels in global search, but can also be used to build more efficient parametric optimisation software by using the eigenvectors as input vectors to the DS-LS-SVM model, while at the same time using the improved ISSA's global search capability and potential parallelism to The DS-LS-SVM is based on the use of the feature vectors as input vectors to the DS-LS-SVM model. The pseudo-code of ISSA-DS-LS-SVM is shown in Algorithm 1.

Algorithm 1 ISSA-DS-LS-SVM

Input: Training data X, labels Y, number of layers L, sparsity parameter λ, regularization parameter γ, kernel function K, number of sparrows N, maximum number of iterations T.

Ouput: Trained model parameters α, b, and sparse features H.

1: Initialize the sparrow population with random positions and velocities.

2: Evaluate the fitness of each sparrow using the mean squared error between the predicted and actual labels.

3: Update the personal best position and fitness for each sparrow.

4: Update the global best position and fitness for the population.

5: Update the velocity and position of each sparrow using the following equations.

a. $v_i\left(t+1\right) = w{\cdot}v_i\left(t\right) + c_1{\cdot}rand(){\cdot}(pbest_i - x_i\left(t\right)) + c_2{\cdot}rand(){\cdot}(gbest - x_i\left(t\right))$

b. $x_i\left(t+1\right) = x_i\left(t\right) + v_i\left(t+1\right)$

6: Update the model parameters α and b using the best position found by the sparrow population.

7: **For** $l = 1$ to L do

a. Compute the kernel matrix K_l using the sparse features H_l.

b. Solve the LS-SVM problem to obtain the parameters α_l and b_l.

c. Compute the residual error $E_l = Y - K_l\alpha_l - b_l$

d. Update the sparse autoencoder weights and biases using backpropagation with E_l as the target.

8: Return the trained model parameters α and b, and the sparse features H.

5. Experimental results and analysis.

5.1. **Experimental design.** To confirm the effectiveness of the ISSA and ISSA-DS-LS-SVM algorithms, the following 2 experiments were conducted in this work:

(1) To determine whether ISSA is superior to the general algorithm, this was achieved primarily by comparison with GA, PSO, ACO, AFSA and SSA. The results of optimisation tests using six standard functions are used, thus demonstrating the effectiveness of ISSA.

(2) To more fully verify the advancedness of the proposed ISSA-DS-LS-SVM-based prediction model and its advantages in defect prediction, this work compares its simulation with DS-LS-SVM, AFSA-DS-LS-SVM and SSA-DLSSVM models. The experimental environment is shown in Table 1. the experimental parameters of ISSA are shown in Table 2.

Table 1. Experimental environment.

Experimental platform	Specific parameters
Memory	16 G
Systems	Centos 7
Framework	Tensorflow, Keras
Language	Python

Table 2. Experimental parameters.

Parameters	Numerical values
Number of layers L	6
Sparsity parameter λ	0.2
Regularization parameter γ	0.6
Kernel function K	Radial basis function
Number of sparrows N	100
Maximum number of iterations T.	20,000

5.2. **ISSA simulation experiments.** The population size was set to 100, the number of iterations was set to 2000 and the dimensionality was set to 30 for all population intelligence algorithms. The expressions of the five test functions were as follows:

$$f_1(x) = \sum_{i=1}^{n} x_i^2, x \in [-100, 100] \tag{23}$$

$$f_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|, x \in [-10, 10] \tag{24}$$

$$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2, x \in [-100, 100] \tag{25}$$

$$f_4(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right], x \in [-5.12, 5.12] \quad (26)$$

$$f_5(x) = 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e, x \in [-32, 32] \quad (27)$$

The optimisation results of each algorithm on the five tested functions are shown in Table 3. The standard deviation of ISSA is the smallest among the results obtained, which indicates that its robustness is also good, and its average and worst optimisation values are also better than those of the other algorithms. Overall, the ISSA algorithm is not only more stable, but also more accurate in its search for the best. In addition, a t-test with a confidence level of 5% was carried out. The results of the t-test show that ISSA does not come from the same distribution as the other algorithms. '-' indicates that it comes from a different distribution and '+' indicates that it comes from the same distribution. Taking

Table 3. Comparison between ISSA and other algorithms

Test functions	Algorithms	Optimal optimization values	Tie-breaker optimisation values	Worst optimized value	Standard deviation	t-test
f1	GA	6.40E+00	6.88E+00	7.48E+00	3.68E-01	-
	PSO	5.56E+01	7.94E+01	9.58E+01	1.28E+01	-
	ACO	4.28E+01	6.82E+01	1.05E+02	2.10E+01	-
	AFSA	1.99E+00	2.48E+00	2.70E+00	2.26E-01	-
	SSA	9.00E+01	1.16E+02	1.29E+02	2.44E+01	-
	ISSA	3.00E-137	1.65E-109	4.38E-109	1.35E-99	None
f2	GA	3.38E+04	4.92E+04	5.82E+04	6.58E+03	-
	PSO	2.96E+04	3.76E+04	4.48E+04	4.64E+03	-
	ACO	3.30E+04	4.60E+04	5.32E+04	6.68E+03	-
	AFSA	2.98E+04	4.38E+04	5.80E+04	1.41E+04	-
	SSA	3.18E+04	4.38E+04	5.80E+04	1.56E+04	-
	ISSA	0.00E+00	0.00E+00	0.00E+00	0.00E+00	None
f3	GA	1.00E+02	5.40E+02	5.64E+02	3.28E+01	-
	PSO	5.02E+02	5.80E+02	6.18E+02	3.30E+01	-
	ACO	2.96E+02	3.90E+02	4.36E+02	4.52E+02	-
	AFSA	2.26E+02	2.56E+02	2.84E+01	1.98E+01	-
	SSA	3.24E+02	3.84E+02	3.98E+02	3.72E+01	-
	ISSA	1.60E+01	2.68E+01	3.90E+01	6.12E+00	None
f4	GA	6.80E+02	9.50E+02	1.12E+03	1.33E+02	-
	PSO	2.58E+02	3.58E+02	4.24E+02	4.94E+01	-
	ACO	2.52E+02	3.76E+02	3.86E+02	7.74E+01	-
	AFSA	3.24E+02	3.81E+02	4.14E+02	3.63E+01	-
	SSA	3.50E+02	3.98E+02	4.28E+02	4.26E+01	-
	ISSA	0.00E+00	0.00E+00	0.00E+00	0.00E+00	None
f5	GA	3.42E+01	3.68E+01	3.92E+01	1.71E+00	-
	PSO	3.58E+01	3.78E+01	3.94E+01	1.11E+00	-
	ACO	3.62E+01	3.76E+01	3.86E+01	7.74E-01	-
	AFSA	3.24E+00	3.60E+00	4.00E+00	2.60E-01	-
	SSA	3.64E+01	3.98E+02	3.72E+01	4.60E-01	-
	ISSA	8.88E-15	8.88E-15	8.88E-15	0.00E+00	None

the test function f3 as an example, the convergence curves of the various algorithms are shown in Figure 3. It can be seen that when the dimension is 30, ISSA converges the fastest compared to the rest of the algorithms, and also has the best accuracy in its search for merit. At the same time, the evolutionary curve of this algorithm can drop in a short time when it first starts, so this means that ISSA has better robustness as it does not impose too many restrictions on the initial position of the artificial fish; algorithms other than ISSA show premature maturity after several executions, so this means that the ISSA algorithm can avoid premature maturity more effectively.

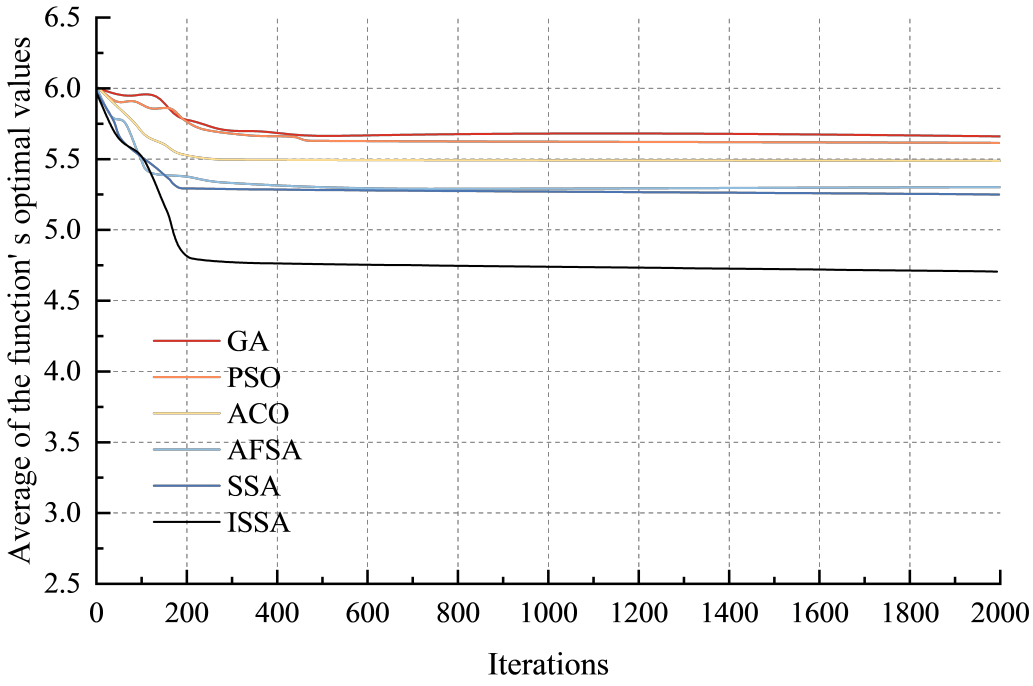

Figure 3. Evolution curve of benchmark function optimization

5.3. **Software defect prediction simulation.** To more fully validate the advancedness of the proposed ISSA-DS-LS-SVM based prediction model and its advantages in defect prediction, this work simulates it against DS-LS-SVM, AFSA-DS-LS-SVM, and SSA-DS-LS-SVM models. Software defect prediction datasets are an important resource for evaluating the performance and effectiveness of machine learning models and algorithms for software defect prediction tasks. Software defect prediction datasets used in the simulation are:

(1) NASA Software Defect Dataset: This dataset, provided by the NASA Software Defect Detection and Prediction Project, contains three years of data from 18 software projects, including pre-processed features and defect markers.

(2) EUSES dataset: This dataset contains 236 Excel spreadsheets and is used to predict whether the cells in them contain errors.

(3) PROMISE dataset: This dataset is provided by the PROMISE software engineering programme and contains data from 20 files from 10 software projects for defect prediction.

(4) KC1 dataset: This dataset contains 210 Java files and is used to predict the presence of defects.

(5) MC1 dataset: This dataset contains 16 C++ programs for predicting defects.

The above datasets are publicly available and can be used in a wide range of software defect prediction studies. In accordance with the metrics commonly used in information mining principles, the confusion matrix metrics are used to compare the evaluation capabilities of the models. Therefore, the confusion matrix shown in Table 4 is used. Using

Table 4. Confusion matrix.

Actual category	**Forecast category (with defects)**	**Forecast category (without defects)**
Module with defects	Correct positive example (TP)	Wrong negative example (FN)
Without defective modules	Positive examples of errors (FP)	Correct negative example (TN)

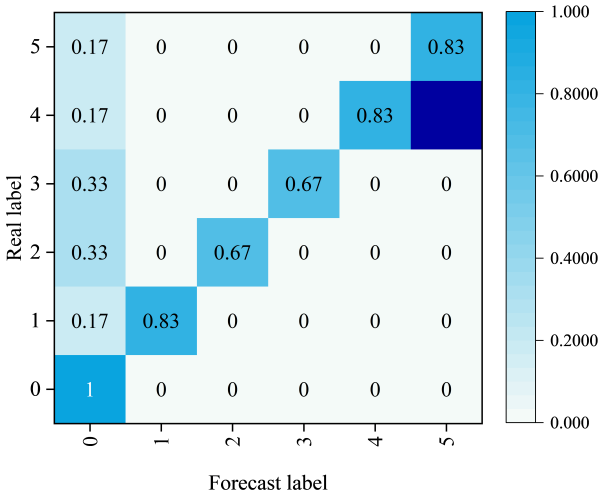

Figure 4. DS-LS-SVM

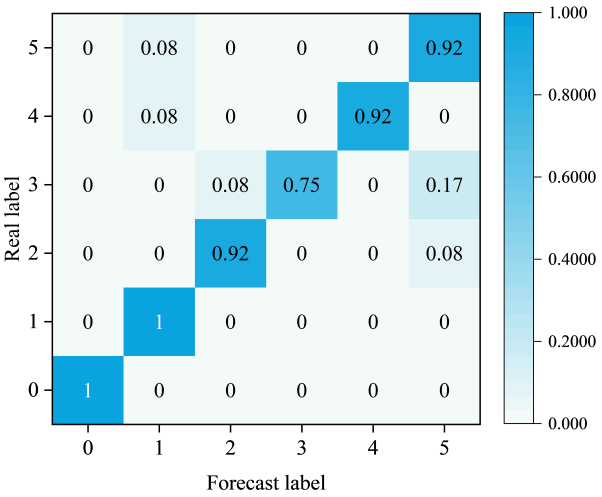

Figure 5. AFSA-DS-LS-SVM

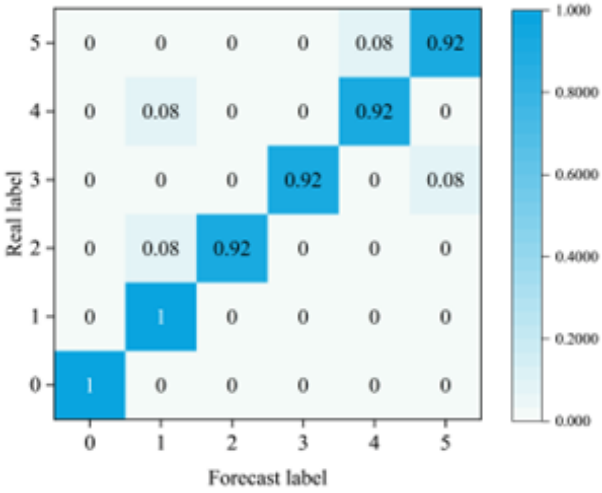

Figure 6. SSA-DS-LS-SVM

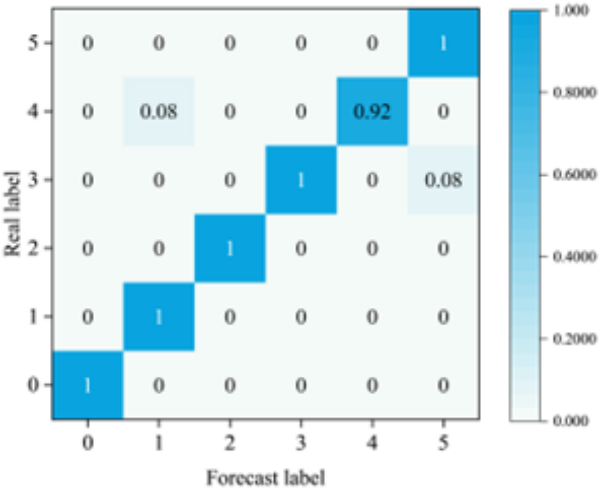

Figure 7. ISSA-DS-LS-SVM

NASA as an example, the confusion matrices for the different models are shown in Figures 4, 5, 6 and 7 respectively, where the horizontal axis represents the predicted defect type number, the vertical axis represents the true defect type number, the diagonal line represents the classification accuracy and the off-diagonal line represents the misclassification rate.

It can be seen that all models correctly determine the state of the software during normal operation, with ISSA-DS-LS-SVM identifying only slightly lower defects for labels 3 and 4, at 92%, corresponding to a false positive rate of 8%. All other types of software defects can be identified accurately. The experimental results verify that the classification performance of ISSA-DS-LS-SVM is significantly better than other models.

6. **Conclusion.** To address the problems of long time and low accuracy of searching the optimal hyperparameter combination of DS-LS-SVM by traditional methods, a prediction model based on ISSA-optimized DS-LS-SVM is proposed, using Circle chaotic mapping to initialize the population, which increases the diversity and stability of the population and expands the search range of the sparrow in space, thus improving the efficiency of the algorithm in finding the optimal; introducing Gaussian differential The algorithm's ability to jump out of the local optimum is improved by introducing Gaussian variance

perturbation at the optimal position to avoid the phenomenon of premature convergence. The results validate the effectiveness of ISSA by comparing experiments on the benchmark function search. Finally, the hyperparameters of DS-LS-SVM are optimised using ISSA to establish an optimal classification model and to conduct comparative experiments with other optimisation-seeking algorithms for software defect prediction, and the results show that the proposed method can accurately discriminate software defects. However, the current classification method with ISSA-DS-LS-SVM prediction model is limited to the division of presence and absence of defects. Could the next step be to consider the use of multi-classification DS-LS-SVM for predicting the severity of software defects, thus providing greater help for software testing.

REFERENCES

[1] K. E. Bennin, A. Tahir, S. G. Macdonell, and J. Brstler, "An empirical study on the effectiveness of data resampling approaches for cross-project software defect prediction," *IET Software*, vol. 16, pp. 112-124, 2022.

[2] T.-Y. Wu, Q. Meng, Y.-C. Chen, S. Kumari, and C.-M. Chen, "Toward a Secure Smart-Home IoT Access Control Scheme Based on Home Registration Approach," *Mathematics*, vol. 11, no. 9, 2123, 2023.

[3] T.-Y. Wu, F. Kong, Q. Meng, S. Kumari, and C.-M. Chen, "Rotating behind security: an enhanced authentication protocol for IoT-enabled devices in distributed cloud computing architecture," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, no. 1, 36, 2023.

[4] T.-Y. Wu, Q. Meng, L. Yang, S. Kumari, and M. Pirouz, "Amassing the Security: An Enhanced Authentication and Key Agreement Protocol for Remote Surgery in Healthcare Environment," *CMES-Computer Modeling in Engineering & Sciences*, vol. 134, no. 1, pp. 317-341, 2023.

[5] F. Zhang, T.-Y. Wu, J.-S. Pan, G. Ding, and Z. Li, "Human motion recognition based on SVM in VR art media interaction environment," *Human-centric Computing and Information Sciences*, vol. 9, no. 1, 40, 2019.

[6] S. Ren, Z. Zhen, L. Yuan, and R. Xie, "Genetic Algorithm-based Transfer Learning for Cross-Company Software Defect Prediction," *International Journal of Hybrid Information Technology*, vol. 10, no. 3, pp. 45-56, 2017.

[7] Shepperd, Martin, Bowes, David, Hall, and Tracy, "Researcher Bias: The Use of Machine Learning in Software Defect Prediction," *IEEE Transactions on Software Engineering*, 2014.

[8] C.-M. Chen, Y. Hao, and T.-Y. Wu, "Discussion of "Ultra Super Fast Authentication Protocol for Electric Vehicle Charging Using Extended Chaotic Maps"," *IEEE Transactions on Industry Applications*, vol. 59, no. 2, pp. 2091-2092, 2023.

[9] C.-M. Chen, L.-L. Xu, K.-H. Wang, S. Liu, and T.-Y. Wu, "Cryptanalysis and Improvements on Three-party-authenticated Key Agreement Protocols Based on Chaotic Maps," *Journal of Internet Technology*, vol. 19, no. 3, pp. 679-687, 2018

[10] C.-M. Chen, K.-H. Wang, T.-Y. Wu, J.-S. Pan, and H.-M. Sun, "A Scalable Transitive Human-Verifiable Authentication Protocol for Mobile Devices," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1318-1330, 2013.

[11] B. Turhan, T. Menzies, A. B. Bener, and J. D. Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, no. 5, pp. 540-578, 2009.

[12] R. S. Wahono, "A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks," *Journal of Software Engineering*, vol. 1, no. 1, pp. 440-453, 2015.

[13] S. Wang and X. Yao, "Using Class Imbalance Learning for Software Defect Prediction," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 434-443. 2013.

[14] X. Yang, T. Ke, and Y. Xin, "A Learning-to-Rank Approach to Software Defect Prediction," *IEEE Transactions on Reliability*, vol. 64, no. 1, pp. 234- 246, 2015.

[15] A. Youssef, "Global-local least-squares support vector machine (GLocal-LS-SVM)," *PLoS One*, vol. 18, no. 4, p. e0285131, 2023.

[16] R. R. Majeed and S. K. D. Alkhafaji, "ECG classification system based on multi-domain features approach coupled with least square support vector machine (LS-SVM)," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 26, no. 5, pp. 540-547, 2023.

[17] Y. Ma, X. Liang, G. Sheng, J. T. Kwok, M. Wang, and G. Li, "Noniterative Sparse LS-SVM Based on Globally Representative Point Selection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 788-798, 2021.

[18] N. Arya, S. Ghordoyee Milan, and Z. Kayhomayoon, "The prediction of longitudinal dispersion coefficient in natural streams using LS-SVM and ANFIS optimized by Harris hawk optimization algorithm," *Journal of Contaminant Hydrology*, vol. 240, p. 103781, 2021.

[19] J. Fan, T. Chen, and S. Lu, "Superpixel guided deep-sparse-representation learning for hyperspectral image classification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3163-3173, 2017.

[20] Y. Wang, Q. Liu, J. Sun, L. Wang, X. Song, and X. Zhao, "Multistrategy Improved Sparrow Search Algorithm Optimized Deep Neural Network for Esophageal Cancer," *Computational Intelligence and Neuroscience*, vol. 2022, p. 1036913, 2022.

[21] C. Zhao, "An Apple Fungal Infection Detection Model Based on BPNN Optimized by Sparrow Search Algorithm," *Biosensors (Basel)*, vol. 12, no. 9, 2022.

[22] X. Ren, S. Chen, K. Wang, and J. Tan, "Design and application of improved sparrow search algorithm based on sine cosine and firefly perturbation," *Mathematical Biosciences and Engineering*, vol. 19, no. 11, pp. 11422-11452, 2022.

[23] F. S. Gharehchopogh, M. Namazi, L. Ebrahimi, and B. Abdollahzadeh, "Advances in Sparrow Search Algorithm: A Comprehensive Survey," *Archives of computational methods in engineering : state of the art reviews*, vol. 30, no. 1, pp. 427-455, 2023.

[24] F. Liu, P. Qin, J. You, and Y. Fu, "Sparrow Search Algorithm-Optimized Long Short-Term Memory Model for Stock Trend Prediction," *Computational intelligence and neuroscience*, vol. 2022, pp. 3680419-3680419, 2022.

[25] C. Zeng, T. Qin, W. Tan, C. Lin, Z. Zhu, J. Yang, and S. Yuan, "Coverage Optimization of Heterogeneous Wireless Sensor Network Based on Improved Wild Horse Optimizer," *Biomimetics (Basel, Switzerland)*, vol. 8, no. 1, 70, 2023.

[26] Y. Ma, X. Liang, G. Sheng, J. T. Kwok, M. Wang, and G. Li, "Noniterative Sparse LS-SVM Based on Globally Representative Point Selection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 788-798, 2021.

[27] X.-L. Xia, "Training sparse least squares support vector machines by the QR decomposition," *Neural Networks*, vol. 106, pp. 175-184, 2018.

[28] S. Nan, L. Sun, B. Chen, Z. Lin, and K.-A. Toh, "Density-Dependent Quantized Least Squares Support Vector Machine for Large Data Sets," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 1, pp. 94-106, 2017.