Journal of Computers Vol. 36 No. 5, October 2025, pp. 315-330
https://doi.org/10.63367/19911599202510360502 1

Application of Improved Firefly Algorithm and Support Vector Machine Fusion Strategy in Chinese Text Classification

Meng-Ying Yang*, Yuan Yuan, Cheng-Fang Mei, and Xiao-Jing Guo

Hebei Institute of Mechanical and Electrical Technology,
Xingtai City 054000, Hebei Province, China
{yangmengying, yuanyuan, meichengfang, guoxiaojing}@hbjd.edu.cn

Received 1 August 2025; Revised 22 September 2025; Accepted 7 October 2025

Abstract. How to quickly and accurately obtain useful information from a large amount of data is an urgent problem that people need to solve. This article focuses on the problems of insufficient search ability and low accuracy of traditional support vector machine parameter optimization methods, and conducts detailed research on optimizing support vector machine parameters to improve the accuracy of text classification and reduce classification time. In order to improve the classification ability of support vector machines, the swarm intelligence optimization algorithm firefly algorithm was introduced when selecting parameters. And improved the firefly algorithm, using the improved firefly algorithm to optimize support vector machine parameters, reducing the blindness of support vector machine parameter selection, and improving the accuracy and speed of text classification. Finally, the performance of the standard support vector machine and the improved firefly algorithm optimized support vector machine in text classification was compared through experiments. The experimental results show that the improved support vector machine model applied in text classification accelerates the classification speed, significantly improves the accuracy of classification, enhances the classification performance of the support vector machine, and verifies the effectiveness of the improved algorithm.

Keywords: text classification, firefly algorithm, support vector machine, parameter search

1 Introduction

With the rapid advancement of information technology, the growth of information is increasing at an exponential rate. The massive growth of information has changed people's lives, but it has also brought difficulties to people, that is, how to quickly and accurately obtain the resources they need in the huge amount of information. Compared to manual information classification methods in the past, automatic classification and filtering of information is a problem that intelligent algorithms need to solve. The basic principle of text classification is to classify text information into one or more predetermined categories based on its content. Automatic text classification can handle a large number of information classification problems and has broad application potential in natural language understanding and processing, information filtering, email processing, and other fields [1].

With the continuous increase in the accumulation scale of network information data, data mining has emerged to identify available resources from a large amount of information. Machine learning based text classification methods are an important part of data mining, mainly studying the patterns in information that cannot be obtained through analytical principles [2]. These patterns are used to analyze objective things and predict unknown data. The important content that needs to be used in the process of data mining is statistical principles. Traditional statistical content is the extreme characteristics exhibited by the sample size when it is extremely large. In the 1990s, proposed a new machine learning method based on statistical theory - Support Vector Machines (SVM). SVM, based on minimizing structural risk, can achieve high classification and recognition accuracy as well as outstanding generalization ability under limited test sample size conditions [3]. By applying the principle of kernel function, nonlinear problems can be transformed into linear problems, reducing algorithm complexity and making it easier to apply to classification problems. SVM has solved the "curse of dimensionality" and overfitting problems, and has been widely applied in facial recognition, image processing, text classification, and other fields. However, SVM also has some drawbacks, as parameters have a significant impact on it and it is difficult to choose appropriate parameters. Nowadays, the dimensionality of the problem is increasing and the complexity is

* Corresponding Author

constantly increasing [4]. Therefore, SVM is an area worthy of in-depth research.

SVM has great advantages in text classification, with fast processing speed and the ability to perform dimensionality reduction, simplifying complex problems. The initial text classification originated from foreign countries, which classified English. Chinese classification started relatively late, with immature knowledge and skills, and Chinese word formation is much more complex than English. However, the number of Chinese users ranks first globally, and with the rapid development of network technology and the trend towards global standardization of resources, the storage capacity of Chinese resources is rapidly increasing [5].

However, in the process of using SVM algorithm to classify Chinese text, there is a defect of sensitivity to parameter selection, which can lead to an increase in the computational complexity of the entire algorithm and an increase in hardware memory consumption. Therefore, this article focuses on accurate classification of Chinese text. Based on the functional defects of SVM basic algorithm, the work done in this article is as follows:

Firstly, this article completed text classification and dataset preparation. Text vectors were constructed and text features were processed. Then, SVM algorithm model was used, and an improved firefly algorithm was used to search for parameters to address the difficulty in determining model parameters. Finally, an experimental environment was set up to verify the effectiveness of the proposed method in this article.

2 Related Work

For the research on text classification methods, commonly used algorithms include Bayesian algorithm, K-nearest neighbor algorithm, neural network, SVM, etc. Leng Ting used the Naive Bayes method to classify news texts, proposed a factor analysis Naive Bayes classification model, and classified news texts in the Fudan University text corpus. At the same time, the results were compared with Naive Bayes, multiple logistic regression models, and decision tree methods. Finally, after experimental verification, the factor analysis Naive Bayes model achieved an accuracy of 84% in classifying Chinese texts.

Jiao Wu [6] from China Jiliang University constructed a linear regression classification model based on class neighborhood dictionaries, using the K-nearest neighbor method to construct class neighborhood dictionaries for each category. Based on different representations of the test samples, she proposed linear regression classification algorithms based on cascaded class neighborhood dictionaries and class neighborhood dictionaries, respectively. At the same time, to alleviate the impact of noisy data on classification performance, noisy class data was cropped by measuring the correlation between the test samples and each category.

The frequency of using SVM for Chinese text classification is the highest. For example, Qichao Zhang [7] from Jiangsu University used support vector machine as the basic classifier and optimized the parameters of firefly algorithm by adding variable step size factor. The adaptability of variable step size factor solved the oscillation and other phenomena of firefly algorithm. Finally, multiple SVM basic classifiers with different weights were iteratively trained using Adaboost framework, and integrated to obtain a stronger classifier with better performance. The algorithm performed well in recall and accuracy.

This article uses SVM model to classify text, but SVM has the problem of difficulty in selecting its own parameters. Therefore, how to improve the efficiency of parameter selection, Xiang Zheng [8] optimizes the support vector machine parameters through an improved empire competition algorithm to improve fault diagnosis performance.

Yichen He [9] from Yanshan University proposed a prediction algorithm for multi kernel support vector machines based on parameter optimization, and used the grey wolf algorithm to determine the parameters of different kernel support vector machines, which improved the progress of the algorithm.

Dongmei Zhao [10] from Hebei Normal University inputs the extracted features into a support vector machine for classification, and uses a grid optimization algorithm to globally optimize the parameters in the support vector machine.

Yuxin Yang [11] from Harbin Institute of Technology optimized the parameters of the support vector machine model using genetic algorithm, which improved the training efficiency and generalization performance of the model. Finally, in numerical and field experiments, the algorithm effectively improved the accuracy of detecting and identifying buried small targets in aeromagnetic exploration.

3 Text Classification and Dataset

Text information is divided into two categories, one is used to train models, known as the training set; The other type is test data, called test sets. Use the categories obtained from the test set to evaluate the accuracy of the classification model. The overall process is shown in Fig. 1:

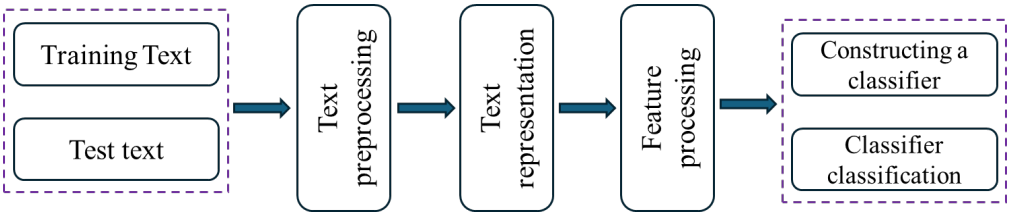

Fig. 1. Overall flowchart of text classification

3.1 Text Preprocessing

Text preprocessing is a critical stage in natural language processing and text mining [12], directly impacting the quality of input features for downstream tasks such as classification, clustering, and sentiment analysis. For Chinese text, preprocessing typically includes three main steps: text annotation and denoising, Chinese word segmentation, and stop word filtering. Text data collected from the internet is rarely clean, as most text data is embedded in web pages and contaminated with non-linguistic noise, such as HTML tags, JavaScript or script content within <script> blocks, and multimedia information such as video, audio, advertisements, emoticons, and embedded images. While these elements are useful for web page display, they do not contribute to semantic analysis and often interfere with classification. Therefore, they must be filtered out during preprocessing, typically through the use of regular expressions, HTML parsers, or rule-based tag removal. After text annotation and denoising, the semantic purity of the data is significantly improved, noise is reduced, and the quality of the text information is enhanced, creating favorable conditions for subsequent word segmentation and feature extraction. Because Chinese doesn't use spaces to separate words, word segmentation is crucial for dividing continuous character sequences into meaningful lexical units. Word segmentation methods vary, including lexicon-based approaches, those relying on maximum forward or backward matching, statistical methods employing probabilistic models, and comprehension-based approaches that mimic human language cognition. This article focuses on comprehension-based approaches that are knowledge-driven and require syntactic and semantic analysis of sentences, resolve polysemy through contextual disambiguation, and leverage knowledge-based reasoning to improve word segmentation accuracy. This approach enables computers to mimic how humans understand language, breaking down sentences in a manner closer to natural understanding and is particularly effective in domains where ambiguity is prevalent. After word segmentation, the resulting tokens often contain some high-frequency words with weak semantics, such as function words, modal words, or structural markers. These words are called stop words. A common approach to filtering out stop words is to construct a stop word list and match the tokens against it. If a token matches an entry in the list, it is classified as a stop word and deleted; if it does not, it is retained as a meaningful feature. Removing stop words reduces dimensionality, eliminates noise, and improves the efficiency and accuracy of text classification models by retaining only semantically relevant words. These three steps of text preprocessing transform noisy raw data into clean, structured, and meaningful input, laying a solid foundation for further natural language processing tasks.

3.2 Text Representation

In typical text collections used for classification, feature words are treated as independent attributes with no inherent correlation between them and are unaffected by their order in the original text [13]. This assumption simplifies the representation of text to a bag-of-words model, where the presence or frequency of a word is important, but its position within a sentence or paragraph does not influence the analysis. Within this framework,

probabilistic models are constructed by treating each feature as an independent variable and performing statistical analysis to estimate the user's potential interest and the probability distribution of text occurrences within the dataset. Probabilistic models work by calculating the probability of each feature word appearing in the entire text collection and then combining these probabilities to form a statistical representation of each document. After determining these probabilities, the results are sorted in descending order of importance or contribution. This sorting process highlights the most representative or influential features, which can serve as strong indicators of the text's likely category. To further optimize the classification process, feature words are assigned different weights based on their ranking in the ordered list. Higher-ranked features are given higher weights, while less informative features are given lower weights. This weighting mechanism ensures that the classifier emphasizes the most relevant features, thereby improving classification accuracy and efficiency. Finally, weighted features are incorporated into a probabilistic model to perform text classification by comparing the likelihood of a given document belonging to each category and assigning the document to the category with the highest probability score. This process strikes a balance between simplicity and effectiveness. By assuming independence between features, the complexity of the probabilistic model is reduced, making it computationally efficient and scalable to large text datasets. Furthermore, by ranking features and applying different weights, the model overcomes the independence assumption, ensuring that more discriminative terms have a greater impact on the classification results. The result is a powerful probabilistic framework that can capture user interests and accurately classify text in large-scale information retrieval and classification tasks.

In the vector space model, the text set D consists of a set of texts (d_1, d_2, ..., d_n), where d_i represents a document in the document set; Each text is composed of multiple feature items (t_1, t_2, ..., t_n), where t_i represents one feature item in the text information. If the weight of the feature item is represented by w_i, then the document can be represented as $d = d\ \{w_1, w_2, ..., w_n\}$, where w_i represents the weight of the feature item t_i, $1 \le i \le n$. The text classification problem is transformed into a computational problem between several vectors, reducing the complexity of text classification. The similarity between text information can be represented by the distance between vectors, and commonly used methods include vector inner product method, Jaccard coefficient method, and cosine angle coefficient method.

The commonly used method is the cosine angle method, which calculates the cosine value of the angle between two vectors using a cosine value table, the higher the cosine value, the greater the similarity between texts. The calculation formula is as follows:

$$sim(D_1, D_2) = \sum_{i=1}^{n} w_{1k} * w_{2k} \tag{1}$$

$$sim(D_1 D_2) = \frac{w_i * w_q}{\sqrt{\sum_{j=1}^{n} w_{ik}^{\ 2}} \sqrt{\sum_{j=1}^{n} w_{jk}^{\ 2}}} \tag{2}$$

In the above equation, w_{ij} represents the weight of the j-th feature of the i-th document.

3.3 Feature Processing

The feature processing stage includes two steps: feature extraction and feature weighting.

1) Feature extraction, this article uses the information gain method for feature extraction, represented as follows:

$$\begin{aligned} IG(w) = &-\sum_{i=1}^{m} P(C_i) \log P(C_i) + P(W) \sum_{i=1}^{m} P(C_i \mid W) \log P(C_i \mid W) \\ &+ P(\overline{W}) \sum_{i=1}^{m} P(C_i \mid \overline{W}) \end{aligned} \tag{3}$$

In the above equation, $P(C_i|W)$ represents the probability of a document belonging to the C_i class when it con-

tains the feature term W, $P(\overline{W})$ represents the probability of a document in the training set that does not contain the feature term W, $P(C_i|\overline{W})$ represents the conditional probability of belonging to the C class when the feature term W is not included in the training set, and m represents the number of categories of the text.

A common method for feature selection in text classification is to calculate the information gain (IG) value for each feature in the training set. Information gain measures a feature's contribution to reducing the uncertainty of the class labels. The higher the IG value, the stronger the feature's classification discriminative ability. After calculating the IG values for all features, a threshold is set: features below this threshold are removed, while features above this threshold are retained. The retained features form a reduced feature space, which serves as the new input representation for training the classifier. This dimensionality reduction process not only improves computational efficiency by eliminating redundant or irrelevant features, but also improves the performance of the classification model by focusing on the most informative attributes. However, using information gain as a feature selection criterion has limitations. A significant drawback is that when a feature appears only in a single class, the calculated IG value is often very small, even if the feature itself is a strong indicator of that class. This is because IG emphasizes the distribution of a feature across all classes. If a feature is highly concentrated in a single class and cannot distinguish between multiple classes, the entropy reduction of the entire dataset will be limited, resulting in an underestimated information gain score. Therefore, although such features may help identify a specific class, they may be discarded during feature selection. This shortcoming can negatively impact classification accuracy, especially in situations with severe class imbalance or the presence of rare but unique features. To address this, researchers sometimes combine information gain with other feature selection methods, such as the chi-square statistic, mutual information, or term frequency-inverse document frequency (TF-IDF) weighting, to more comprehensively capture the discriminative power of a feature. Another approach is to apply hybrid or multi-criteria selection strategies to ensure that infrequent but class-specific features are not discarded simply because of a low information gain score.

2) Feature weighting assigns weights to feature items extracted during the feature extraction process based on their importance. The higher the frequency of appearance of statistical feature items in text, the greater the significance of feature items for text classification. The higher the Inverse Document Frequency (IDF) value, the more concentrated the feature items appear, indicating a greater discriminative effect on the text. The calculation method is as follows:

$$idf(T_k) = \log\left(\frac{N}{n_k} + 0.01\right) \tag{4}$$

The formula for calculating TF-IDF weights based on the above three factors is as follows:

$$w(t,d) = \frac{tf(t,d) \times \log\left(\frac{N}{n_k} + 0.01\right)}{\sqrt{\sum_{t \in d}\left[f(t,d) \times \log\left(\frac{N}{n_k} + 0.01\right)\right]^2}} \tag{5}$$

Among them, $w(t, d)$ represents the weight of the feature item, $tf(t,d)$ represents the frequency of the feature item appearing in the text, N is the total number of texts, and the denominator is normalization processing.

4 SVM Strategy Based on Improved Firefly Algorithm

4.1 Construction of SVM Model

Support vector machine has great advantages in solving small sample and high-dimensional vector spaces [14]. For linearly separable optimal classification surfaces, it not only requires correctly separating the two types of problems, but also requires the maximum classification interval. When linearly inseparable, a hyperplane cannot completely separate the two types of samples, and relaxation variables can be introduced. In this case, the opti-

mal classification surface obtained is the generalized optimal classification hyperplane. The general function of linear discrimination is expressed as:

$$g(x) = w^T x + b = 0 \tag{6}$$

The equation for the optimal classification surface is expressed as $w^T x + b = 0$. The points that satisfy the function are:

$$y_i[w \cdot x_i + b] - 1 \geq 0, i = 1, 2, \cdots, n \tag{7}$$

The classification surface that satisfies the above equation and maximizes the value of $2/||w||$ is the optimal hyperplane. The optimal classification surface problem can also be expressed as a constrained optimization problem, where the function is solved under the constraints of equation (7):

$$\varphi(w) = \frac{1}{2}\|w\|^2 = \frac{1}{2}(w^T w) \tag{8}$$

The minimum value problem in the above equation can also be transformed into a dual problem, which is generally solved by introducing the Lagrange function, as shown in the following equation:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{i} \alpha_i (y_i (w_i + b) - 1) \tag{9}$$

In the above equation, $\alpha_i = (\alpha_i, ..., \alpha_n)^T$ is the Lagrange multiplier and $\alpha_i \geq 0$ is the Lagrange coefficient. To obtain the minimum value of the function, perform partial differentiation on w, b, and a_i to make them equal to 0. The above problem can be transformed into a dual problem of convex quadratic programming:

$$\begin{aligned} &\min_{\alpha} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i x_j) - \sum_{j=1}^{l} \alpha_j \\ &s.t \sum_{i=1}^{l} y_i \alpha_{\ i} = 0, \alpha_i \geq 0, i = 1, 2, \cdots l \end{aligned} \tag{10}$$

The solution $a^* = \left(a_1^*, \cdots, a_i^*\right)^T$ obtained from the above problem has some sample points corresponding to each value of a_i^*. The sample points obtained by $a_i^* > 0$ are the support vectors. The optimal solution to the initial problem can be obtained, and the decision function for the optimal classification surface is as follows:

$$\begin{cases} f(x) = \text{sgn}(\sum_{i=1}^{l} \alpha_i^* y_i (x \cdot x_i) + b^*) \\ b^* = y_j - \sum_{i=1}^{l} \alpha_i^* y_i (x \cdot x_i) \end{cases} \tag{11}$$

According to the above equation, the decision function is determined by the support vector and is independent of the sample points where a_i^* equals 0.

When linearly inseparable, if a hyperplane does not have the property of separating two types of samples, a relaxation variable $\xi_i (\xi_i \geq 0, i = 1, 2, ..., l)$ can be added to make the optimal classification surface satisfy:

$$y_i (w^T x_i + b) \geq 1 - \xi_i \tag{12}$$

The optimal hyperplane obtained at this point is the generalized optimal classification hyperplane. When

$0 < \xi_i < 1$, sample point x_i can be classified correctly, while when $\xi_i > 1$, sample point x_i is not classified correctly. Therefore, the following objective function is required:

$$\psi(w,\xi) = \frac{1}{2} w^T w + C\sum_{i=1}^{n} \xi_i \tag{13}$$

In the formula, C is the normal value and is called the penalty factor. SVM needs to allocate a cost function in the original problem, introduce a penalty term C, and transform it into the following objective function:

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i \\ &s.t. y_i\left(w \cdot x_i + b\right) \geq 1 - \xi_i \\ &\xi_i \geq 0, i = 1,2,...,l \end{aligned} \tag{14}$$

Among them, this method is suitable for cases where the samples are approximately linearly separable. For complex sample classification problems, complex metasurfaces need to be used instead of classification hyperplanes. It is solved through the following dual problem:

$$\begin{aligned} &\min_{\alpha} \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k\left(x_i \cdot x_j\right) - \sum_{j=1}^{l} \alpha_j \\ &s.t. \sum_{i=1}^{l} y_i \alpha_i = 0 \\ &0 \leq \alpha_i \leq C, i = 1,...,l \end{aligned} \tag{15}$$

SVM transforms the training samples into a high-dimensional space through a mapping function φ, making them linearly separable in the feature space, and then constructs the optimal classification hyperplane. The nonlinear transformation is implemented through a kernel function, which is represented as follows:

$$k(x_i \cdot x_j) = \phi(x_i) \cdot \phi(x_j) \tag{16}$$

Among them, "·" represents inner product. If $k(x_i, x_j)$ is a semi positive definite kernel, then the problem is a convex quadratic programming. The difference between linearly inseparable support vector machine and linearly separable support vector machine is that the value of a_i is limited. If the value of a_i^* is greater than zero, then x_i is called a support vector; If the value of a_i^* is equal to C, then x_i is called the boundary support vector; If the value of a_i^* is between zero and C, then xi is called an in bound support vector. The specific solutions a^* and b^* are obtained from the above equation, and then obtained through a^* and kernel function. Finally, the decision function is obtained:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i * y_i k\left(x_j, x_i\right) + b*\right) \tag{17}$$

The optimal classification hyperplane is determined by samples with $a^* > 0$ and is independent of others. Radial basis function (RBF) is a scalar function that is radially symmetric, and Gaussian kernel function is the most commonly used function. Its form is as follows:

$$k(x, x_i) = \exp\left\{-\frac{\| x - x_i \|^2}{2\sigma^2}\right\} \tag{18}$$

The optimal classification hyperplane is determined by samples with $a^* > 0$ and is independent of others. In the above equation, xi is the center of the function, representing the distance between a certain point and the core

point, and σ is the width parameter, representing the control of the radial range.

When $x = 0$, the image of the radial basis kernel classifier is shown in Fig. 2.

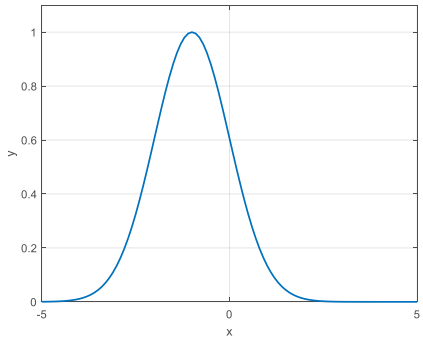

Fig. 2. Radial basis function classifier image

4.2 Optimizing SVM Parameters Using Improved Firefly Algorithm

SVM has shown good performance in classification recognition accuracy and generalization ability when the number of training samples is limited, and it also has certain advantages in dealing with nonlinear and high-dimensional space problems [15]. However, SVM is sensitive to the selection of kernel functions and parameters, and the performance of the algorithm is greatly affected by the set parameters. The parameter selection problem of SVM is a process of parameter optimization. The algorithm optimization process is as follows:

1) Initialize various parameters, including the number of fireflies m, the maximum fluorescence intensity attraction β_0, the light intensity absorption coefficient γ, the step size factor a_0, and the searchable range $[x_{min}, x_{max}]$; The initial position of the firefly and the maximum number of iterations n. The range of values for the X-axis is $[C_{min}, C_{max}]$, and the range of values for the Y-axis is $[\sigma_{min}, \sigma_{max}]$.

2) Calculate the initial values of penalty factor C and kernel parameter σ based on the libsvm package. The optimization objective is to minimize the classification error of SVM, defined as

$$f(C,\sigma) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, h(x_i; C, \sigma)) \tag{19}$$

3) Based on the updated position of fireflies, recalculate their fluorescence brightness and attractiveness to determine their movement direction. The attractiveness between fireflies decreases exponentially with distance:

$$\beta = \beta_0^{e-\gamma r^2} \tag{20}$$

$$r = \sqrt{\sum_{i=0}^{d} (x_i - y_i)^2} \tag{21}$$

4) Update the firefly position based on the improved FA and perturb the optimal firefly position. To balance exploration and exploitation, the step size is adjusted dynamically as follows:

$$\alpha_t = \alpha_0 (1 - \frac{t}{t_{max}}) \tag{22}$$

5) Calculate the brightness of fireflies after the update.

6) When the maximum search count is reached or the search accuracy is met, proceed to the next step;

Otherwise, if the search frequency increases, return to step (3) and search again.

7) Output global extremum and optimal solution, program ends.

In order to balance the search ability and convergence speed during the algorithm evolution process, the original fixed α value was changed, and α was transformed. The iteration number variable was added to the step size factor, and α was adjusted to a linearly decreasing value. As the number of iterations increased, the step size gradually decreased, avoiding the phenomenon of unstable step size. The improved firefly algorithm in this paper is referred to as IFA [16]. The adjusted α is as follows:

$$\alpha = \alpha_0 * \frac{\left\| x_i - x_{best} \right\|}{d_{\max}} * \frac{n}{i} \tag{23}$$

In the formula, α_0 is the initialization step factor, x_{best} is the optimal position of the firefly, $||x_i - x_{best}||$ represents the spatial distance between the current firefly and the optimal firefly, which can be represented by Euclidean distance; $d_{\max}$ represents the maximum distance between the optimal firefly individual and neighboring fireflies.

After updating the formula, in the early stage of the algorithm, the value of α increases compared to before, and the search range of the step size also increases, enhancing the search ability; As the number of iterations increases, α decreases linearly. In the later stages of the algorithm, the search step size gradually decreases and the convergence speed increases, avoiding the problem of slow convergence speed in later stages of the algorithm. In order to avoid the search step size being too large or too small, and to ensure that the updated position of the firefly is within the set search range, the step size range is limited. For firefly positions that exceed the search range, the following formula is used:

$$x_i \begin{cases} x_{\min}, x_i < x_{\min} \\ x_{\max}, x_i > x_{\max} \end{cases} \tag{24}$$

When the search length is less than or equal to the minimum value, the minimum value of the search range is taken as the step size. When the search step size is greater than or equal to the maximum value, the maximum value of the search range is taken as the step size to ensure that the step size is not too large or too small, and to conduct the search efficiently. The algorithm flow is shown in Fig. 3.

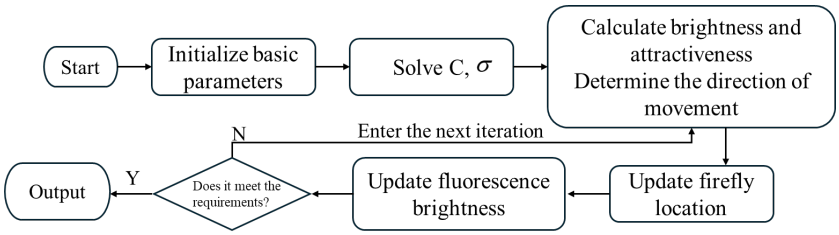

Fig. 3. Algorithm flowchart

The pseudocode for the algorithm section is as follows:

```
INPUT:
    - Training_dataset: {x_i, y_i}, i = 1, 2, ..., n
    - Max_Generations: Maximum number of iterations
    - n: Number of fireflies in the population
    - α: Randomness strength parameter (0 < α < 1)
    - β0: Initial attractiveness at distance r=0
    - γ: Light absorption coefficient (controls the decrease in attractiveness with distance)
    - lower_bound: Lower bound for the parameters [C_min, gamma_min]
    - upper_bound: Upper bound for the parameters [C_max, gamma_max]
```

```
  OUTPUT:
       - best_firefly: The firefly with the highest brightness (best solution
found)
       - best_fitness: The corresponding best fitness value (e.g., highest
classification accuracy)
  BEGIN
      // Step 1: Initialization
       // Initialize a population of n fireflies randomly within the search
space
      population = InitializePopulation(n, lower_bound, upper_bound)
      // Evaluate the initial fitness (brightness) of each firefly
      FOR i = 1 to n DO
          // A firefly's position represents a candidate solution (C, gamma)
          C = population[i].position[0]
          gamma = population[i].position[1]
           // Train an SVM model with parameters (C, gamma) on the training
data
          svm_model = train_svm(Training_dataset, C, gamma)
           // Evaluate the model, typically using cross-validation accuracy
as fitness
          // Higher accuracy means higher brightness (I)
            population[i].fitness = evaluate_fitness(svm_model, Training_data-
set)
          population[i].brightness = population[i].fitness
      END FOR
```

5 Simulation Experiment and Result Analysis

5.1 Experimental Process Design

The experimental software is Matlab, and the development package used is LIBSVM, a pattern recognition and regression software package developed. This experiment was conducted in a Windows environment, so the built-in compiler of the development package can be modified to meet the experimental standards.

The data format of the libsvm package in the experiment is as follows:

<label><index>:<value1><index2>:<vaule2>…

<label>tags represent category identifiers, usually in integer form; The values of<index>start from 1 and are all integers; <value>is the training data, which is the feature value in the classification data.

The classification dataset used in this article also comes from Sogou Laboratory [17]. Since 2006, it has been manually edited and organized, repeatedly supplemented and updated, and the dataset has rich categories and professional classification. There are 10 categories in the corpus, each of which contains 7999 documents. This paper selects five categories of economy, politics, environment, Internet and culture for the experiment, of which 1000 are selected for each category, 700 for the training set, and 300 for the test set.

In the corpus, the category folders are named with different numbers, and the text format is TXT file. The file names are named with numerical numbers, and the documents in each category are numbered starting from the number 10 to ensure fairness in classification. The document format and text examples are shown in Fig. 4.

Convert the dataset required for the experiment into the format required by MATLAB and import it into MATLAB. Choose the type of kernel function, and in this experiment, RBF [18] is chosen as the kernel function for the support vector machine. Select the kernel function parameters for the improved firefly algorithm. Train the obtained parameters on the training set. Classify the text using the obtained support vector machine model.

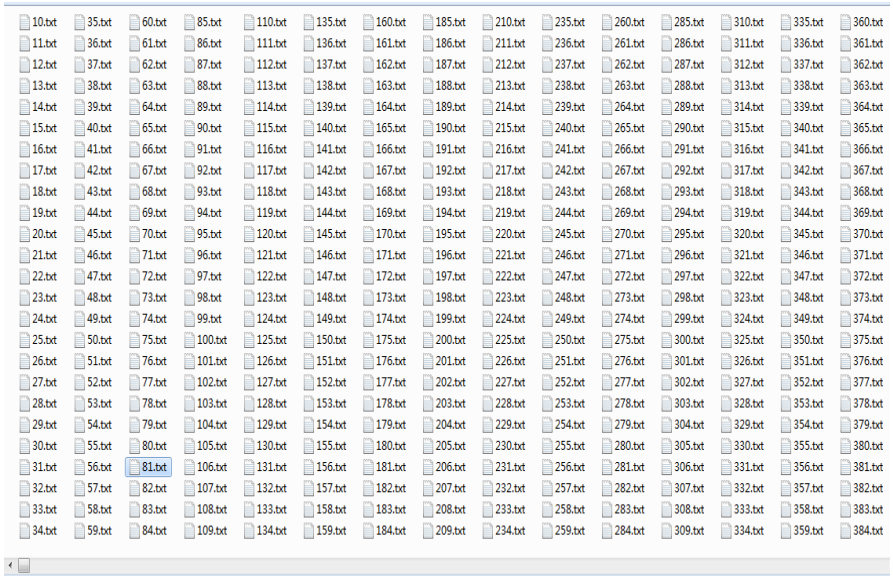

Fig. 4. Text format

5.2 Experimental Results and Analysis

To verify the performance of the improved firefly algorithm, Ackley function, Sphere function, and Rosenbrock function were selected for simulation testing [19]. The population size is set to 25, the light absorption coefficient γ is 1, the maximum fluorescence brightness attractiveness β_0 is 2, the step size factor α_0 is 0.2, the dimension is set to 10, and the maximum number of iterations is set to [100, 1000] depending on the function. Table 1 shows the selected function expressions.

Table 1. Comparative description between classic functions for testing algorithm performance

Function name	Function expression	Value range	Optimal solution
Ackley	$f(x)=-20e^{(-0.2\sqrt{\frac{1}{10}\sum_{i=1}^{n}x_i^2})}-e^{\frac{1}{10}\sum_{i=1}^{n}\cos 2\pi x_i}+20+e$	[-100,+100]	0
Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$	[-100,+100]	0
Rosenbrock	$f(x)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	[-30,30]	0

1) In the simulation experiment of Ackley function, in order to verify the effectiveness of the improved algorithm in this paper, the iteration number was set to 500 times, and each algorithm experiment was conducted 10 times. The final data obtained was averaged, and the experimental data is shown in Table 2.

Table 2. Ackley function data table

Algorithm name	Total times	Total time (s)	Optimal solution times	Optimal solution
FA	500	16.011340	500	8.8818e-15
IFA	500	4.078083	62	2.2204e-15

Two algorithms were tested on the Ackley function in MATLAB, and two search process curves were obtained. The curves were slightly different each time, and the more stable curve was selected, as shown in Fig. 5.

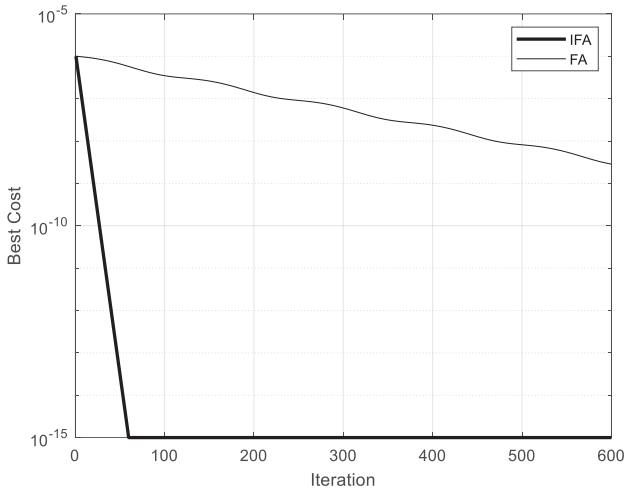

Fig. 5. Comparison chart of Ackley function curves

2) Sphere function simulation experiment, in order to verify the performance of the improved algorithm in this paper, the iteration times were set to 1000, and each algorithm experiment was conducted 10 times. The final data obtained was averaged, and the experimental data is shown in Table 3.

Table 3. Sphere function data table

Algorithm name	Total times	Total time (s)	Optimal solution times	Optimal solution
FA	1000	18.561770	1000	6.6725e-21
IFA	1000	13.571364	644	0

Two algorithms were tested on the Sphere function in MATLAB, and two search process curves were obtained. The selected curve is the more stable one, as shown in Fig. 6.

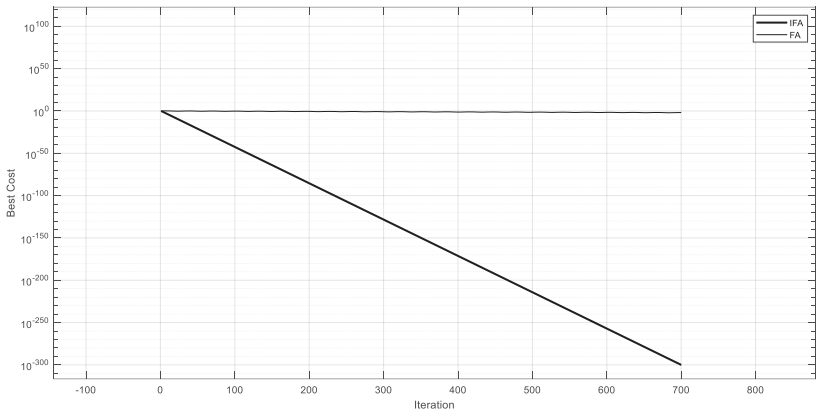

Fig. 6. Comparison chart of Sphere function curves

3) Rosenbrock function simulation experiment, in order to verify the effectiveness of the improved algorithm in this paper, the iteration times were set to 500, and each algorithm experiment was conducted 10 times. The final data obtained was averaged, and the experimental data is shown in Table 4.

Table 4. Rosenbrock function data table

Algorithm name	Total times	Total time (s)	Optimal solution times	Optimal solution
FA	500	15.654240	500	8.9534e-07
IFA	500	10.217560	500	4.7037e-08

Two algorithms were tested on the Rosenbrock function in MATLAB, and two search process curves were obtained. The selected curve is the more stable one, as shown in Fig. 7.

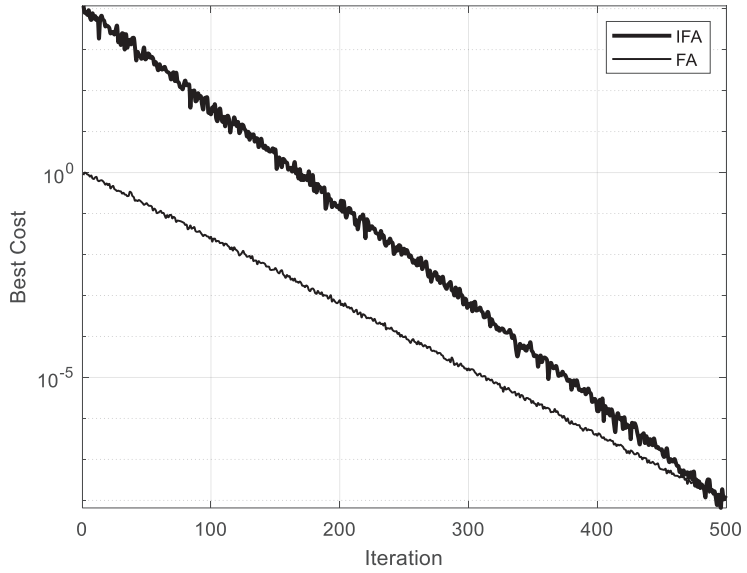

Fig. 7. Comparison chart of Rosenbrock function curves

According to the table and comparison curve of the simulation experiment, Tabular results and simulation comparison curves demonstrate significant performance differences between the improved Firefly Algorithm (IFA) [20] and the standard Firefly Algorithm (FA) [21] on various benchmark functions. For the Ackley function, the IFA converges quickly, reaching the global optimal solution in just 62 iterations, demonstrating its powerful ability to escape local minima and quickly find the global optimum. In contrast, the standard FA's convergence curve is relatively flat and slow. Even after 500 iterations, the FA still fails to reach the optimal solution, indicating its limited global search capability and slow convergence. Furthermore, the IFA is computationally efficient, with a search time approximately one-quarter that of the FA. This demonstrates that the proposed improved method not only improves solution quality but also significantly reduces computational cost. For the Sphere function, the IFA's advantage is again evident, converging to the optimal solution after 644 iterations, while the FA fails to reach the same optimal solution even after 1000 iterations. This result demonstrates that the IFA possesses superior convergence efficiency and accuracy for unimodal optimization problems, making it more effective at finding the optimal solution than the traditional FA. For the Rosenbrock function, known for its narrow valley structure and optimization difficulty, both algorithms ultimately achieved very close optimal solutions. However, the comparison curves show that IFA converges faster in the initial stages, quickly approaching the optimal solution and stabilizing with increasing iterations. In contrast, FA converges more slowly, requiring more iterations to achieve comparable results. This demonstrates that even for complex and challenging functions, IFA offers significant advantages in accelerating optimization. The fundamental reason for IFA's improved performance lies in its dynamic step-size adjustment strategy. In the initial stages of the algorithm, we intentionally increase the search step size to enhance global search capabilities, enabling the algorithm to more broadly explore the search space and avoid premature convergence to local minima. As iterations increase, the step size decreases linearly, shifting the search focus to fine-tuning and local exploitation. This adaptive mechanism accelerates convergence in the later stages, ensures algorithm stability, and enables

the algorithm to efficiently and accurately obtain the optimal solution. Overall, experimental results on multiple benchmark functions fully demonstrate the superiority of the improved Firefly algorithm. By strengthening global exploration in the early stages and optimizing local utilization in the later stages, the improved Firefly Algorithm (IFA) achieves faster convergence, higher accuracy, and less computational time than the traditional Firefly Algorithm. This demonstrates the effectiveness and practicality of the improved algorithm and highlights its potential application in solving various complex optimization problems.

2) Compare and analyze the experimental results between the improved SVM classifier (I-SVM) and the standard SVM classifier to verify the feasibility and efficiency of the proposed method in this paper. The classification effect is compared by measuring the precision (P), recall (R), and F-measurement value (F1) of the statistical document.

The parameter settings of the improved firefly include a light absorption coefficient γ of 1, an attraction degree β_0 of 2 for maximum fluorescence brightness, a maximum iteration number of 300, and a step size factor α_0 of 0.2. In order to improve the accuracy and authenticity of the experiment, when selecting σ and C, each category was averaged 5 times to obtain the average value, which was then applied to text classification. The experimental results are shown in Table 5.

Table 5. Parameter values

Algorithm name	Total times	Total time (s)	Optimal solution times	Optimal solution
FA	500	15.654240	500	8.9534e-07
IFA	500	10.217560	500	4.7037e-08

According to Table 5, the values of the penalty factor C in I-SVM are all larger than those in SVM. The values of C differ by 200-300 for the five categories of economy and culture. The main function of C is to adjust the proportion of experience risk and improve promotion ability. The relatively large value of C in I-SVM indicates that the greater the experience penalty, the smaller the experience risk, which ensures both the promotion ability and the proportion of risk of SVM; On the contrary, in each category, the σ value of RBF in I-SVM is smaller than that in SVM. The main function of σ is to control the radial range, and the smaller the value of σ, the more support vectors there are, resulting in more correct classifications. However, the value of σ is controlled within a certain range, ensuring the learning ability of the classifier.

The comparison chart of precision, recall, and F1 before and after algorithm improvement is shown in Fig. 8.

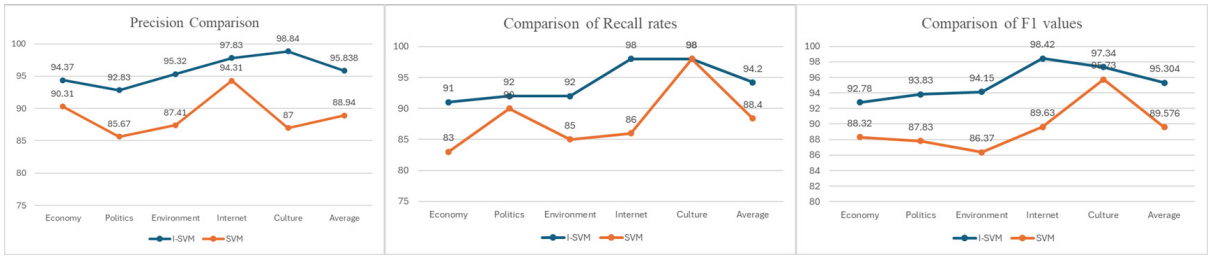

Fig. 8. Performance comparison before and after algorithm improvement

As shown in Fig. 8, the proposed I-SVM algorithm significantly outperforms the traditional SVM across all key evaluation metrics, including precision, recall, and F1 score. Specifically, in the cultural text classification task, the I-SVM achieves 11.84% higher precision than the traditional SVM, and in the internet text classification task, the I-SVM achieves 12% higher recall and 8.79% higher F1 score. These improvements demonstrate that the enhanced model consistently outperforms the baseline model, and the I-SVM achieves higher precision, recall, and F1 score than the standard SVM. These results confirm that parameter-optimized SVMs using the improved Firefly algorithm can significantly improve both classifier accuracy and robustness. This strongly suggests that combining the improved Firefly algorithm with SVMs offers an effective and feasible solution for improving text classification performance. In addition to improvements in accuracy-related metrics, the efficiency of the classification process is also enhanced. The I-SVM achieves significantly faster classification time than the traditional SVM. For example, in an ambient text classification task, the I-SVM reduced processing

time by approximately 50 seconds compared to a baseline method. This reduction in computational cost highlights the dual advantages of this approach: it not only improves classification accuracy, recall, and overall performance, but also accelerates the classification process, bringing practical benefits in real-world applications, which often involve large-scale and time-sensitive data. The improved Firefly algorithm plays a decisive role in parameter optimization and plays an important role in achieving a balance between accuracy and efficiency. By systematically searching the parameter space and avoiding local optima, the improved Firefly algorithm is able to find optimal SVM parameters, thereby improving classification accuracy and reducing computational overhead. When these optimized parameters are subsequently applied to a text classification task, the classifier achieves faster convergence, more efficient resource utilization, and shorter training and testing times. Overall, these results confirm that the proposed I-SVM approach is not only theoretically sound but also has practical application value. This method effectively improves the classification accuracy of support vector machines while reducing classification time, providing an efficient, scalable, and reliable solution for text classification. This validates the feasibility of the method and demonstrates its potential for application in various fields where both accuracy and speed are crucial.

6 Conclusion

This article mainly introduces the research background and significance of text classification, the general process of text classification, and related theories and technologies. The main content of this article is as follows:

(1) A comparison was made between the commonly used algorithms SVM, Naive Bayes algorithm, and K-nearest neighbor algorithm in text classification. Through research and experiments, the results showed that SVM is the most effective algorithm in text classification.

(2) The firefly algorithm has been introduced, which has a simple structure, is easy to operate, and has fewer adjustable parameters. However, the firefly algorithm itself also has some shortcomings. Improvements have been made to the firefly algorithm, and experiments have verified that the improved firefly algorithm has enhanced performance, improved early search ability, and accelerated convergence speed in the later stage.

(3) The traditional SVM parameter selection method, although simple in principle, has a relatively large computational cost. This article applies the improved firefly algorithm to SVM parameter optimization and applies the optimized parameters to train the SVM model.

(4) Through experimental comparison, the experimental results verified that the improved SVM algorithm applied to text classification has higher classification accuracy and speed than the standard SVM, which validates the feasibility of the proposed new algorithm.

At the same time, there are also some research shortcomings in this article, which are also future research directions:

(1) The improved algorithm in this article is based on the firefly algorithm. Although the firefly algorithm has a simple structure, it requires the support of multiple software during the experimental process, which increases the burden on the computer. Therefore, a more convenient and efficient method for selecting SVM parameters should be developed.

(2) Due to the use of PC computers in the experiment, the number of datasets processed is limited, making it impossible to conduct experiments on a large number of datasets. In reality, datasets are huge, so better hardware facilities are needed to handle large-scale data problems.

(3) Feature processing is the core step of text classification, but the current algorithm structure is complex and computationally intensive. Further research should be conducted on better feature processing methods to improve the efficiency of text classification.

7 Acknowledgement

Research on Anomaly Intrusion Detection Methods and Protection Mechanisms Based on Industrial Internet of Things Network Security (2025ZC030).

References

[1] S. Gao, S.-J. Li, Z.-P. Cai, A survey of Chinese text classification based on deep learning, Computer Engineering & Science 46(4)(2024) 684-692.

[2] X.-M. Liu, C.-Z.-X. Li, S.-C. Wu, Y.-C. Zhang, H.-Y. Bai, Z.-H. Cheng, Z. Chen, Y.-F. Li, Y. Lan, C. Shen, A Survey of Text Classification Algorithms and Application Scenarios, Chinese Journal of Computers 47(6)(2024) 1244-1287.

[3] C.-M. Jia, Y.-Y. Xiao, T. Jiang, Review of granular support vector machines, Journal of Tianjin University of Technology 40(3)(2024) 57-66.

[4] S.-F. Ding, Y.-T. Sun, Z.-Z. Liang, L.-L. Guo, J. Zhang, X. Xu, Survey on Support Vector Machine Algorithms in Weakly Supervised Scenarios, Chinese Journal of Computers 47(5)(2024) 987-1009.

[5] T. Leng, R.-Y. Ye, Y.-J. Li, News Text Classification Based on Naive Bayesian Method of Factor Analysis, Journal of Anqing Normal University (Natural Science Edition) 30(1)(2024) 47-51.

[6] J. Wu, C.-F. Hong, Y.-C. Gu, X.-Q. Gu, S.-J. Jin, Linear Regression Text Classification Based on Class-wise Nearest Neighbor Dictionary, Computer Engineering 47(8)(2021) 93-99+108.

[7] Q.-C. Zhang, L.-Y. Zhou, L.-C. Ding, Research on Improvement of Short Text Classification Algorithm Based on MRMR and SVM, Computer & Digital Engineering 53(1)(2025) 164-169.

[8] X. Zheng, H.-H. Zhang, Y.-Y. Wei, Research on infrared image fault diagnosis method for substation equipment based on improved SVM, Electrical Measurement & Instrumentation 61(12)(2024) 49-55.

[9] Y.-C. He, C.-L. Shi, X.-Q. Guo, W. He, T. Han, Photovoltaic Power Prediction Algorithm Based on Parameter Optimization of Multi-Kernel SVM, Acta Energiae Solaris Sinica 45(9)(2024) 394-404.

[10] D.-M. Zhao, M.-W. Sun, M.-Y. Su, Y.-X. Wu, Network Security Situation Assessment Based on Improved SKNet-SVM, Journal of Applied Sciences 42(2)(2024) 334-349.

[11] Y.-X. Yang, P.-F. Zhang, B.-J. Li, Y. Shen, Z.-Y. Chen, X.-B. Yang, A Study on Weak Magnetic Anomaly Detection Algorithm of Target Based on Improved Support Vector Machine, Modern Radar 45(12)(2023) 62-68.

[12] Y.-L. Su, W.-J. Li, X.-Y. Liu, J.-P. Ding, S.-X. Liu, H.-N. Li, G.-F. Li, Review of Text Classification Methods Based on Graph Neural Networks, Computer Engineering and Applications 60(19)(2024) 1-17.

[13] Y.-F. Deng, Y.-K. Feng, S.-Y. Wang, Multidimensional semantic extract method for emergency decision-making texts: a technical framework based on TF-IDF and PMI, Journal of Safety Science and Technology 21(5)(2025) 36-45.

[14] Z.-J. Gu, X.-Y. Yang, H. Sui, Improved SVM Model for Industrial Control Anomaly Detection Based on Interference Sample Distribution Optimization, Computer Science 52(7)(2025) 388-398.

[15] D. Liu, W.-L. Xing, Y.-Y. Zhang, H.-J. Chen, C.-S.-Y. Li, L. Lv, Firefly Algorithm with Vertical Horizontal Crossover and Spiral Movement, Jiangxi Science 43(2)(2025) 321-329.

[16] J.-X. Gu, X.-S. He, X.-S. Yang, Improved firefly algorithm optimizes twin support vector machine parameters, Microelectronics & Computer 39(11)(2022) 11-18.

[17] X.-Y. Chen, J.-J. Qi, X.-M. Zhu, X. Wang, Z. Wang, Unlabelled text mining methods based on two extension models of concept lattices, International Journal of Machine Learning and Cybernetics 11(2)(2020) 475-490.

[18] Q.-Z. Liu, C. Chen, Y. Zhang, Z.-G. Hu, Feature selection for support vector machines with RBF kernel, Artificial Intelligence Review 36(2011) 99-115.

[19] A.-A. Shahri, M.-K. Zak, H.-A. Shahri, A Modified Firefly Algorithm Applying on Multi-objective Radial-based Function for Blasting, Neural Computing and Applications 34(2022) 2455-2471.

[20] G.-G. Wang, L.-H. Guo, H. Duan, H.-Q. Wang, A New Improved Firefly Algorithm for Global Numerical Optimization, Journal of Computational and Theoretical Nanoscience 11(2)(2014) 477-485.

[21] A.H. Gandomi, X.-S. Yang, S. Talatahari, A.H. Alavi, Firefly Algorithm with Chaos, Communications in Nonlinear Science and Numerical Simulation 18(2013) 89-98.