Regression-Based Hyperparameter Learning for Support Vector Machines

Shili Peng, Wenwu Wang, *Senior Member, IEEE*, Yinli Chen, Xueling Zhong, and Qinghua Hu, *Senior Member, IEEE*

***Abstract*— Unification of classification and regression is a major challenge in machine learning and has attracted increasing attentions from researchers. In this article, we present a new idea for this challenge, where we convert the classification problem into a regression problem, and then use the methods in regression to solve the problem in classification. To this end, we leverage the widely used maximum margin classification algorithm and its typical representative, support vector machine (SVM). More specifically, we convert SVM into a piecewise linear regression task and propose a regression-based SVM (RBSVM) hyperparameter learning algorithm, where regression methods are used to solve several key problems in classification, such as learning of hyperparameters, calculation of prediction probabilities, and measurement of model uncertainty. To analyze the uncertainty of the model, we propose a new concept of model entropy, where the leave-one-out prediction probability of each sample is converted into entropy, and then used to quantify the uncertainty of the model. The model entropy is different from the classification margin, in the sense that it considers the distribution of all samples, not just the support vectors. Therefore, it can assess the uncertainty of the model more accurately than the classification margin. In the case of the same classification margin, the farther the sample distribution is from the classification hyperplane, the lower the model entropy. Experiments show that our algorithm (RBSVM) provides higher prediction accuracy and lower model uncertainty, when compared with state-of-the-art algorithms, such as Bayesian hyperparameter search and gradient-based hyperparameter learning algorithms.**

***Index Terms*— Hyperparameter optimization, maximum margin classification, regression, support vector machine (SVM).**

I. Introduction

AMONG the classification algorithms, the maximum margin algorithm is an important type of machine learning methods [1], which can be used for uncertain data processing [2], unsupervised structured prediction [3], multidimensional classification [4], Bayesian network classifier [5], etc. Its typical representative is support vector machine (SVM) [6]. SVM is based on the Vapnik–Chervonenkis dimension theory and obtains the best generalization ability by searching for the largest classification margin in the feature space [7]. It has a solid theoretical foundation and is easy to implement, so it has been successfully applied to a variety of learning tasks [8], such as medical diagnosis [9], image processing [10], and thermographic fault diagnosis [11], [12].

SVM has been developed continuously since it was proposed. A variety of new SVM algorithms are emerging to improve their training speed and promote their applications. Among them, twin SVM (TWSVM) is particularly prominent [13]. For binary-classification problems, TWSVM constructs a hyperplane for each class sample. The sample of each class is as close as possible to the hyperplane of its class and as distant as possible from the hyperplane of the other class [14]. TWSVM not only maintains the advantages of SVM but also improves the training speed by 4× [13]. To further improve the performance of TWSVM and promote its application, researchers have proposed various improved algorithms [13]. For example, the least-squares TWSVM (LSTSVM) is proposed, whose training process is to solve linear equations to achieve faster training speed. To meet the needs of large-scale learning, Ganaie et al. [15] proposed large-scale fuzzy LSTSVM, which does not involve matrix inversion, and thus further improves the training speed.

Noise is ubiquitous in real-world learning tasks, which usually reduces the classification accuracy of the model and makes the learning task more complex [16], [17]. There are also several improved TWSVM and SVM methods for processing noisy data. For example, Tanveer et al. [18] proposed an intuitionistic fuzzy weighted TWSVM, which can reduce the influence of outliers and noise. Moslemnejad and Hamidzadeh [19] applied the belief function theory to the detection of noise and outliers. Hamidzadeh and Moslemnejad [20] used confidence function and rough set theory to determine boundary samples and noise. For multiclass data classification, Moslemnejad and Hamidzadeh [21] proposed weighted SVM to improve the noise sensitivity. For uncertain data, Liang and Zhang [22] proposed uncertainty-aware TWSVM, where an interesting theorem is derived by transforming the multidimensional integrals into 1-D integrals to obtain a simplified model.

Manuscript received 1 July 2021; revised 1 January 2023, 3 September 2023, and 11 September 2023; accepted 29 September 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61925602 and in part by the Key Project of the Humanities and Social Sciences Research in Universities of Guangdong Province under Grant 2018WZDXM032. *(Corresponding author: Shili Peng.)*

Shili Peng, Yinli Chen, and Xueling Zhong are with the School of Internet Finance and Information Engineering, Guangdong University of Finance, Guangzhou 510521, China (e-mail: shilipeng@gduf.edu.cn; cyl@gduf.edu.cn; tzhongxl@gduf.edu.cn).

Wenwu Wang is with the Centre for Vision, Speech and Signal Processing, University of Surrey, GU2 7XH Guildford, U.K. (e-mail: w.wang@surrey.ac.uk).

Qinghua Hu is with the College of Intelligence and Computing, Tianjin University, Tianjin 300072, China (e-mail: huqinghua@tju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3321685.

Digital Object Identifier 10.1109/TNNLS.2023.3321685

However, some key issues that still exist in SVM have not been resolved, such as the choice of hyperparameters [23]. SVM is very sensitive to hyperparameters, and its performance depends on the choice of hyperparameters, such as the bandwidth σ of the Gaussian kernel and the penalty coefficient C [24]. Therefore, the optimization of hyperparameters in SVM has received a lot of attention [25]. Recently, with the focus on complex machine learning models with many hyperparameters [26], the research on hyperparameter optimization algorithms has regained widespread attention [27], [28]. Common methods include grid search, random search, and Bayesian optimization [29]. These algorithms can be directly used for SVM hyperparameter learning.

When the number of hyperparameters is small, grid search is a commonly used optimization method [30]. The grid search method is considered a traditional hyperparameter optimization method, which creates models for each combination of hyperparameters and evaluates their performance on a validation set. Then, the hyperparameter with the smallest error on the validation set is regarded as the optimal hyperparameter. The search range of hyperparameters is generally selected based on experience or adjusted gradually. Although this method is very simple, the number of performance evaluations increases exponentially with the increase in the hyperparameter dimension [31]. Another problem is that when the accuracy of SVM is increased, the number of evaluations for hyperparameters required will be greatly increased [32]. As a result of these factors, the grid search method is inefficient and unable to accommodate large-scale data processing.

To address the limitation in grid search, Bergstra and Bengio [33] proposed a random search method, which usually can obtain better hyperparameters. This method defines a marginal distribution for each hyperparameter, and then randomly selects values to form a set of hyperparameters for model training and validation. In the grid search, if a hyperparameter has no effect on the performance of the model, then the same performance would be obtained even if this hyperparameter is set to different values, provided that the other hyperparameters remain fixed. This leads to many meaningless performance evaluations; however, the random search method can avoid this problem [34]. In addition, random search does not require any assumptions about the learning task. With sufficient computing resources, the random search method can be infinitely close to the globally optimal solution [35]. Grid and random search are relatively simple, but neither of them uses models and historical information. Different from these two methods, the Bayesian hyperparameter optimization method uses historical information to approximate the model and iteratively searches for the optimal hyperparameter [36]. There are two key components in the iterative process: the surrogate model and the acquisition function [32]. Surrogate models are used to fit historical hyperparameters. The acquisition function uses a probability model to predict the performance of different candidate hyperparameters [36]. The Bayesian hyperparameter optimization method has achieved excellent performance in image classification and speech recognition based on deep neural networks [36]. However, the Bayesian hyperparameter optimization method is not always stable, sometimes the performance is excellent, but sometimes catastrophic errors occur on certain tasks [37].

Another important type of hyperparameter learning method is based on the generalization error gradient, which is used to guide the optimization of hyperparameters. Different from black-box model optimization methods such as grid search, random search, and Bayesian search, this method uses the gradient information of the hyperparameters, not just the error information. In practice, the gradient-based hyperparameter learning method is superior to the black-box model-based method in terms of performance and speed. The main reason is that the gradient-based learning method makes full use of the gradient information of the model [37]. Although grid search, random search, and Bayesian search methods can all be used to optimize the hyperparameters of SVM, our research mainly focuses on gradient-based hyperparameter learning to minimize the generalization error of SVM.

The SVM generalization error is a function of the minimum sphere radius and the classification margin in the feature space [7]. When the feature mapping function (i.e., kernel function and hyperparameters) is given, the minimum sphere radius of the sample is fixed. Therefore, SVM can minimize the generalization error by maximizing the classification margin. However, the change in hyperparameters will cause the feature space to change, and the minimum sphere radius is no longer fixed. Therefore, the influence of the minimum sphere radius needs to be considered in hyperparameter learning. For example, Chapelle et al. [38] used the radius-margin (RM) bound as the generalization error estimation to guide SVM hyperparameter learning. At present, most SVM hyperparameter learning is based on RM or its improved algorithm [39]. However, the RM bound is derived from the Vapnik–Chervonenkis dimension, considering the worst case in the feature space. If there are outliers in the sample, it will affect the accuracy of the generalization error estimation [40].

The motivation of this work is to use the leave-one-out method to solve hyperparameter learning, probability estimation, and uncertainty measurement of a maximum margin classifier. Leave-one-out is an unbiased estimation of generalization error for the regression model, but it cannot be directly used for generalization error estimation for classification. Therefore, we convert the maximum margin classification problem into a piecewise linear regression task. Different from the RM methods, we propose a regression-based SVM (RBSVM) hyperparameter learning method. In the leave-one-out generalization error estimation, the loss function must be smooth to calculate its gradient. Therefore, the 0–1 step loss function in classification needs to be transformed into a smooth loss function. A common solution is to approximate the 0–1 step loss function with the sigmoid function $g(x) = (1 + \exp(-ax))^{-1}$ and then calculate its gradient, as in the span algorithm [38]. However, the choice of the constant a is not trivial. If a is too small, the error estimate is not accurate. If a is too large, the error estimate is not smooth, and the samples far away from the discriminant hyperplane will lead to gradient disappearance.

The RBSVM algorithm we proposed is a completely different method, which performs generalization error estimation and hyperparameter learning from a regression perspective. We formulate the SVM classification task as a piecewise linear regression task, use the tanh function to approximate this regression function, and calculate the cross-entropy generalization error. Then the hyperparameter learning is performed according to the cross-entropy generalization error, instead of calculating the step loss of each sample. In addition, the value range of the tanh function is $[-1, +1]$, corresponding to the classification label $\{-1, +1\}$, which can be easily converted into classification probability to measure the uncertainty of the model.

In summary, the contributions of this article can be summarized as follows.

1) We convert the maximum margin classification problem into a regression task and propose a new hyperparameter optimization algorithm RBSVM. The cross-entropy generalization error is applied to SVM generalization error estimation and hyperparameter learning. Converting classification tasks into regression tasks provides a new solution to classification problems, and hyperparameter learning can be considered as its typical application.
2) To achieve an unbiased estimation of the generalization error using the leave-one-out method, we use the tanh function to approximate the piecewise linear function, which is equivalent to the maximum margin classifier, instead of using the sigmoid function to approximate the 0–1 step loss function of each sample. The approximation method aims to make the gradient of the tanh function at the classification boundary equal to the gradient of the piecewise linear function. The reason for choosing this approximation is because the boundary is crucial to the classification model.
3) We propose to use model entropy to measure the uncertainty of the classification model. The model entropy is based on the leave-one-out method, which considers the predicted probability and distribution of all the samples, not just the support vectors. Thus, it offers advantages over the classification margin, which is affected by the regular hyperparameters, does not consider the distribution of the samples, and thus cannot accurately describe the uncertainty of the model.

II. Related Work

It is usually expected that a machine learning model (such as SVM) can select an optimal hyperparameter to minimize the actual prediction error. However, the actual prediction error is often not directly measured and can only be estimated indirectly by calculating the generalization error bound. Therefore, this section introduces the SVM hyperparameter learning method and generalization error estimation, such as leave-one-out error estimation (T_{Loo}), span error estimation (T_{Span}), and RM error estimation (T_{RM}) [38].

A. Support Vector Machine

For the given n training samples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x}$ and y are, respectively, the sample and its corresponding label, SVM maps $\boldsymbol{x}$ to a high-dimensional feature space through the mapping function Φ, and then finds the linear discriminant function with the largest classification margin in the feature space $f(\boldsymbol{x})$ [7]

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \Phi(\boldsymbol{x}) + b \tag{1}$$

where $\boldsymbol{w}$ is the weight coefficient, and b is the bias term.

We mainly study the binary-classification task, namely, $y_i \in \{-1, +1\}$. It is the basic problem of classification tasks, and multiclassification tasks can be converted into multiple binary-classification tasks [41]. In a binary-classification task, if the sample is linearly separable in the feature space, the parameters $\boldsymbol{w}$ and b of the linear discriminant function $f(\boldsymbol{x})$ can be obtained by solving the following convex optimization problem [7]:

$$\begin{aligned} &\min_{\boldsymbol{w},b} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ &\text{s.t. } y_i(\boldsymbol{w}^T\Phi(\boldsymbol{x}) + b) \geq 1 \quad \forall i. \end{aligned} \tag{2}$$

To avoid the dimensionality catastrophe caused by feature mapping $\Phi(\boldsymbol{x})$, the optimization problem (2) is usually not solved directly. Using the Lagrangian dual method, the above optimization problem is transformed into a dual problem to be solved. In duality, the vector inner product is calculated by the kernel function, that is, $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j)$. The duality corresponding to the above original problem (2) is

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \quad \forall i \end{aligned} \tag{3}$$

where α_i is a nonnegative dual variable, and $k(.,.)$ is a specific kernel function. In SVM, the kernel function and its hyperparameters are very important. The most commonly used kernel function is the Gaussian kernel k_{Gau}, which is defined as follows:

$$k_{\text{Gau}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2\sigma^2}\right), \quad \sigma \in R^+ \tag{4}$$

where σ is the hyperparameter of the kernel function. The Gaussian kernel function has good performance and is widely used in various fields.

If the sample is not linearly separable in the feature space, slack variables can be introduced to form a soft margin SVM. There are two types of penalty for slack variables: $L1$-norm and $L2$-norm. For example, the original problem corresponding to the $L2$-norm soft margin SVM is

$$\begin{aligned} &\min_{\boldsymbol{w},b} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 \\ &\text{s.t. } y_i(\boldsymbol{w}^T\Phi(\boldsymbol{x}) + b) \geq 1 - \xi_i \quad \forall i \end{aligned} \tag{5}$$

where C is the hyperparameter that penalizes the training error, and ξ_i is the slack variable [7]. The corresponding dual

problem is

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{1}{C} \right)$$
$$\text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \quad \forall i. \tag{6}$$

From the comparative analysis of $L2$-norm soft margin SVM and hard margin SVM, it can be seen that the soft margin SVM algorithm with the $L2$-norm penalty is equivalent to the hard margin SVM with modified kernel matrix $\widetilde{K}$ [38]

$$\widetilde{K} = K + \frac{1}{C} I \tag{7}$$

where K is the kernel matrix corresponding to the kernel function $k(\cdot, \cdot)$, I is the identity matrix, and C is the penalty coefficient of the slack variables. Since the soft margin SVM can be converted to the hard margin SVM by (7), the derivation and calculation of the generalization error in this article mainly focus on the hard margin SVM.

B. RM Generalization Error Estimation

RM generalization error estimation is a very important error estimation method in SVM. It is derived from the Vapnik–Chervonenkis dimension by Vapnik [7]. RM generalization error upper bound T_{rm} is

$$T_{\text{rm}} = \frac{1}{n} \frac{R^2}{M^2} \tag{8}$$

where M is the maximum classification margin, R is the minimum sphere radius that contains all the samples in the feature space, and n is the number of training samples. Later, Keerthi et al. [42] gave a more accurate estimation. There is a certain constant c, for which the generalization error estimation bound of SVM T_{svm} holds with the probability of $1 - \delta$

$$T_{\text{svm}} = \frac{m}{n} + \sqrt{\frac{c}{n} \left(\frac{R^2}{M^2} \log^2(n) + \log\left(\frac{1}{\delta}\right) \right)} \tag{9}$$

where n is the number of samples in the training set, and m is the number of misclassified samples. If the distribution of the sample in the feature space is a flat ellipsoid, the corresponding sphere radius will be very large, which will affect the accurate estimation of the generalization error. To address this problem, it is usually necessary to normalize the sample to render a uniform spherical distribution in the feature space [38].

C. Leave-One-Out Generalization Error Estimate

The leave-one-out method is another generalization error estimation method, which is an extreme case of K-fold cross-validation. A test sample is selected from the training set, and the remaining $(n-1)$ samples are used to train the model. The model is then used to predict the test sample. Each sample is selected in turn as a test sample, and the remaining samples are used to train the model. The average prediction accuracy of n samples is used as an estimate of model generalization error, which is an almost unbiased estimate. The definition of estimated T_{loo} is as follows [38], [43], [44]:

$$T_{\text{loo}} = \frac{1}{n} \sum_{i=1}^{n} E_i^{n-1}(\boldsymbol{x}_i, y_i) = \frac{1}{n} \sum_{i=1}^{n} \Psi\left(-y_i f^i(\boldsymbol{x}_i)\right) \tag{10}$$

where $E_i^{n-1}(\boldsymbol{x}_i, y_i)$ is the prediction error of $\boldsymbol{x}_i$ by the model trained after removing the sample $\boldsymbol{x}_i$, and $f^i(\boldsymbol{x}_i)$ is its prediction value. $\Psi(x)$ is a nonsmooth step function, when $x > 0$, $\Psi(x) = 1$, otherwise $\Psi(x) = 0$.

The leave-one-out method is a very important generalization error estimation. However, this method needs to build n models to predict the reserved single sample, which is very difficult in practice. Therefore, this method is usually used to analyze and derive other easy-to-calculate generalization error estimation bounds [38].

Opper and Winther [45] were inspired by the linear response theory. Assuming that the set of support vectors does not change after the i sample is removed, the following formula holds:

$$y_i(f^0(\boldsymbol{x}_i) - f^i(\boldsymbol{x}_i)) = \frac{\alpha_i}{\left(K_{\text{SV}}^{-1}\right)_{ii}} \tag{11}$$

where f^0 is the model obtained by training all the samples, and K_{SV} is the support vector kernel matrix. Therefore, the corresponding leave-one-out generalization error estimation bound is

$$T_{\text{loo}} \leq \frac{1}{n} \sum_{i=1}^{n} \Psi\left(\frac{\alpha_i}{\left(K_{\text{SV}}^{-1}\right)_{ii}} - 1 \right). \tag{12}$$

Later, Chapelle et al. [38] considered the change in the support vector set caused by removing a single sample and smoothed the generalization error estimation. Furthermore, they proposed the span error estimation and gave a geometric explanation for it.

D. Span Generalization Error Estimation

Span generalization error estimation bound is derived by Chapelle et al. [38] based on the support vector span. The span value S_i^2 of the support vector $\boldsymbol{x}_i$ is defined as the distance between the sample $\boldsymbol{x}_i$ and the set Λ_i in the feature space. The definition of the set Λ_i is

$$\Lambda_i = \left\{ \sum_{j \neq i, \alpha_j \geq 0} \lambda_j \Phi(x_j), \sum_{j \neq i} \lambda_j = 1 \right\} \tag{13}$$

where α_j is the dual variable corresponding to the sample $\boldsymbol{x}_j$. Correspondingly, the estimation bound of span generalization error is [38]

$$T_{\text{span}} = \frac{1}{n} \sum_{i=1}^{n} \Psi(\alpha_i S_i^2 - 1) \tag{14}$$

where S_i is the span value of the sample $\boldsymbol{x}_i$, and Ψ is the step loss function.

Span is a generalization error estimate that is very close to the leave-one-out method, but the estimate is not continuous [38]. There are two main reasons for this. One is that the Ψ function is a discontinuous step function and needs to be

approximated by a smooth function. The second is that the change in the support vector set will cause the noncontinuous change in Λ_i. One solution is to penalize the change in Λ_i to make it as smooth as possible.

III. Regression-Based SVM

Since the classification label $y_i \in \{-1, +1\}$ is a noncontinuous discrete value, its generalization error estimate is usually also discontinuous. To use the gradient-based hyperparameter learning method, the generalization error estimate needs to be smoothed. The common method is to use the sigmoid function to approximate the 0–1 step loss function.

Different from these existing algorithms, we perform SVM hyperparameter learning from the perspective of regression and propose an RBSVM hyperparameter learning algorithm RBSVM. RBSVM converts the SVM into a piecewise linear regression problem, and then performs hyperparameter learning without artificially setting the approximate function parameters.

A. Regression Equivalent to SVM and Approximation

Nonlinear separable L2-SVM can be converted into linear separable L2-SVM by modifying the kernel matrix. That is, through Formulation (7), the soft margin SVM can be converted into hard margin SVM, so the regression equivalent analysis of SVM only focuses on hard margin SVM classification. L2-SVM can be equivalently converted into a piecewise linear regression. The analysis and proof are as follows.

Theorem 1: The $L2$ regularization SVM classification can be equivalently transformed into a piecewise linear least-square regression with $L2$ regularization.

That is, the L2-SVM classification algorithm

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\Phi(\boldsymbol{x}) + b) \geq 1 - \xi_i \quad \forall i \tag{15}$$

can be equivalently converted into least-square regression

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^{N}(y_i - f_z(\boldsymbol{w}^T\Phi(\boldsymbol{x}) + b))^2 \tag{16}$$

where C is the penalty coefficient of the training error, and $f_z(\cdot)$ is the piecewise linear function

$$f_z(t) = \begin{cases} +1, & \text{if } t > 1 \\ t, & \text{if } -1 \leq t \leq 1 \\ -1, & \text{if } t < -1. \end{cases} \tag{17}$$

Proof: The theorem can be proved separately according to the value of $\boldsymbol{w}^T\Phi(\boldsymbol{x_i}) + b$.

For nonsupport vectors, they are all outside the positive or negative boundary. When $\boldsymbol{w}^T\Phi(\boldsymbol{x_i}) + b > 1$, we know that $f_z(\boldsymbol{w}^T\Phi(\boldsymbol{x_i}) + b) = 1$. In this case, the label is $y_i = 1$, and it can be seen that $y_i - f_z(\boldsymbol{w}^T\Phi(\boldsymbol{x}) + b)) = 0$. That is, the square of the loss in the regression is zero, which is equivalent to the case of $\xi_i = 0$ in L2-SVM.

When $\boldsymbol{w}^T\Phi(\boldsymbol{x_i}) + b < -1$, there is $f_z(\boldsymbol{w}^T\Phi(\boldsymbol{x_i}) + b) = -1$. At this time, the label is $y_i = -1$, and the squared loss in the regression is also zero, which is also equivalent to the case of $\xi_i = 0$ in L2-SVM.

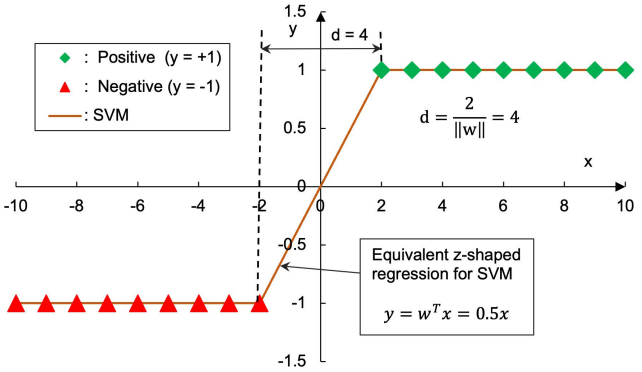

Fig. 1. L2-SVM classification is equivalent to regularized piecewise linear least-squares regression. The red triangle is the negative class, and the green diamond is the positive class. The optimal SVM interface is the y-axis (that is, $x = 0$). The discriminant function is $y = \boldsymbol{w}^T\boldsymbol{x} = 0.5\,x$. The classification margin is $d = 2/||\boldsymbol{w}|| = 4$. The margin is equal to the regularization of the coefficients of the piecewise linear function at the interface, which is $y = 0.5x$.

For support vectors, these samples are located within or on the classification boundary. Therefore, when $-1 < \boldsymbol{w}^T\Phi(\boldsymbol{x_i}) + b < 1$, the squared loss in the regression is $(\boldsymbol{w}^T\Phi(\boldsymbol{x_i}) + b - y_i)^2$, which is equivalent to $(y_i(\boldsymbol{w}^T\Phi(\boldsymbol{x_i}) + b) - 1)^2$ in L2-SVM. That is the case of $\xi_i \neq 0$ in L2-SVM.

The regular term in piecewise linear regression corresponds to the classification margin in L2-SVM. Based on the above discussion, it can be obtained that the L2-SVM classification is equivalent to the regularized least-squares piecewise linear regression. The proof is complete. □

This theorem provides a new perspective for maximum margin classification. The generalization error estimation and uncertainty measurement of the classification task can be analyzed from the perspective of regression. That is, the nonsupport vectors in L2-SVM correspond to samples with zero squared loss in piecewise linear regression, and the support vectors correspond to samples with nonzero squared loss in regression. It should be clarified that L2-SVM is equivalent to piecewise linear regression without losing the excellent sparsity properties of SVM. This is because the purpose of converting L2-SVM into regression is to calculate an unbiased estimate for the generalization error using the leave-one-out method. Then, the learning of hyperparameters is implemented based on this generalization error estimation, whereas the training of the model still uses the L2-SVM. In other words, the RBSVM algorithm is for hyperparameter learning rather than parameter learning. Therefore, after obtaining the optimal hyperparameters, the L2-SVM model with the hyperparameters can still maintain its sparsity property. The equivalent regression of L2-SVM can be more intuitively understood from Fig. 1, where the value of the label y is plus or minus 1, and the classification margin d is equal to the regularization of piecewise linear regression.

Although the maximum margin classification can be equivalent to a piecewise linear regression, the piecewise linear function is not differentiable at the inflection point, and its gradient cannot be directly calculated for parameter learning. Therefore, we use the smooth tanh function to approximate the

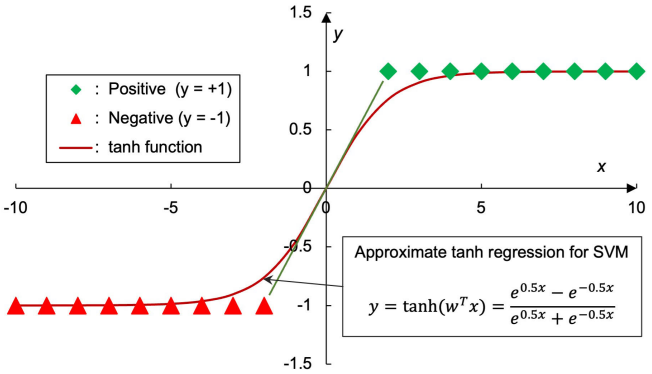

Fig. 2. Continuous regression function $\tanh(0.5x)$ is used to approximate the SVM in Fig. 1. It should be noted that this method does not use the sigmoid function to approximate the 0–1 step loss function, but uses the tanh function to approximate the SVM model.

piecewise linear function. In this way, on one hand, it is convenient to calculate the gradient to optimize the generalization error. On the other hand, the sample classification probability can be obtained according to the tanh function so that the uncertainty of the model is measured (i.e., the model entropy proposed later in this section). The approximation method is to make the gradient of the tanh function at the discriminant intersection (that is, $y = 0$) equal to the gradient of the piecewise linear function at this point. The tanh function is

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}.$$

Its derivative is $\nabla_z \tanh(z) = 1 - y^2$, where y is the corresponding function value. When the independent variable z is $\boldsymbol{w}^T\boldsymbol{x} + b$, the gradients of the tanh function to $\boldsymbol{w}, b$ are

$$\begin{aligned} \nabla_{\boldsymbol{w}} \tanh(\boldsymbol{w}^T\boldsymbol{x} + b) &= (1 - y^2)\boldsymbol{x} \\ \nabla_b \tanh(\boldsymbol{w}^T\boldsymbol{x} + b) &= (1 - y^2). \end{aligned} \tag{18}$$

In the feature space, the function value of SVM at the interface is 0 (i.e., $y = 0$), the derivative with respect to $\boldsymbol{w}$ is $\boldsymbol{x}$, and the derivative with respect to b is 1. Therefore, the tanh function approximated to the piecewise function is

$$g(\boldsymbol{w}, b) = \tanh(\boldsymbol{w}^T\boldsymbol{x} + b) \tag{19}$$

where the values of $\boldsymbol{w}$ and b are the coefficients and bias terms of the SVM, respectively. For example, the tanh function similar to the SVM in Fig. 1 is $g(\boldsymbol{w}, b) = \tanh(0.5x)$, as shown in Fig. 2.

The purpose of using regression to approximate SVM is for hyperparameter learning and uncertainty analysis. On one hand, tanh is a smooth function that facilitates the calculation of the leave-one-out generalization errors and their gradients. The leave-one-out method provides an unbiased estimation of the generalization error, which can accurately reflect the generalization performance of the model [38], [46]. On the other hand, the range of the tanh function is $[-1, +1]$, which can be mapped to the $[0, 1]$ interval to represent the probability of classification. This makes it easier to analyze model uncertainty.

B. Generalization Error Estimation of RBSVM

The hyperparameter learning of RBSVM that we propose here is different from the traditional hyperparameter learning based on the leave-one-out method. The leave-one-out hyperparameter learning method uses leave-one-out to estimate the generalization error after solving L2-SVM. Then, the sigmoid function is used to approximate the 0–1 loss function to make the estimation smooth, and the generalization error gradient is used to guide hyperparameter learning. The RBSVM algorithm is different. After obtaining the L2-SVM optimal solution, RBSVM converts the SVM model into a regression task equivalently and approximates it with the tanh function. Then, the gradient of the cross-entropy loss function is used to guide hyperparameter learning. The generalization error estimation process of RBSVM is shown in Fig. 3.

It should be noted that after SVM classification is converted into a regression task, the corresponding 0–1 loss is converted into a cross-entropy loss, so there is no need to approximate the 0–1 step loss function. The cross-entropy loss is directly used to guide the learning of hyperparameters and the analysis of model uncertainty. That is, traditional L2-SVM classification uses hinge loss, so model analysis and hyperparameter learning need to use 0–1 discontinuous loss function. However, RBSVM uses regression function to approximate SVM, and model analysis and hyperparameter learning are processed by the cross-entropy loss.

When the nonsupport vector (that is, $\alpha = 0$) is removed from the training set, it does not affect the optimal solution. In other words, after the nonsupport vector is removed, it can still be correctly predicted. Therefore, the estimation of generalization error only needs to consider the support vector.

For hard margin L2-SVM, all the support vector points (i.e., $\alpha_i \neq 0$) are on the boundary. According to the KKT dual complementarity condition

$$\alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i) = 0 \tag{20}$$

we can obtain $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i = 0$, that is, $\boldsymbol{w}^T\boldsymbol{x}_i + b + y_i\xi_i = y_i$. In addition, the following equations can be obtained according to the KKT conditions:

$$\boldsymbol{w} = \sum_{j=1}^{N} y_j\alpha_j k(., \boldsymbol{x}_j), \quad \xi_i = \frac{1}{C}\alpha_i, \quad \sum_{j=1}^{N} y_i\alpha_i = 0. \tag{21}$$

Therefore, the KKT condition can be equivalent to the solution of the following linear equation:

$$\begin{bmatrix} \boldsymbol{H} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\alpha}_y \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix} \tag{22}$$

where $H = K + (1/C)I$, and $(\boldsymbol{\alpha}_y)_i = \alpha_i * y_i$.

In leave-one-out, removing nonsupport vectors does not affect the optimal solution, but removing support vectors will affect the optimal solution. Therefore, the analysis of the leave-one-out mainly focuses on the removal of support vectors. For the convenience of presentation, assume that $\boldsymbol{x}_1$ is the removed support vector, and α^{-1} and b^{-1} represent the optimal solution of SVM after removing the support vector $\boldsymbol{x}_1$. To analyze the effect of removing the support vector on the prediction, the following matrix decomposition can be performed [43]:

$$\begin{bmatrix} \boldsymbol{H} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} = \begin{bmatrix} m_{11} & \boldsymbol{m}_1^T \\ \boldsymbol{m}_1 & M_1 \end{bmatrix} = M \tag{23}$$

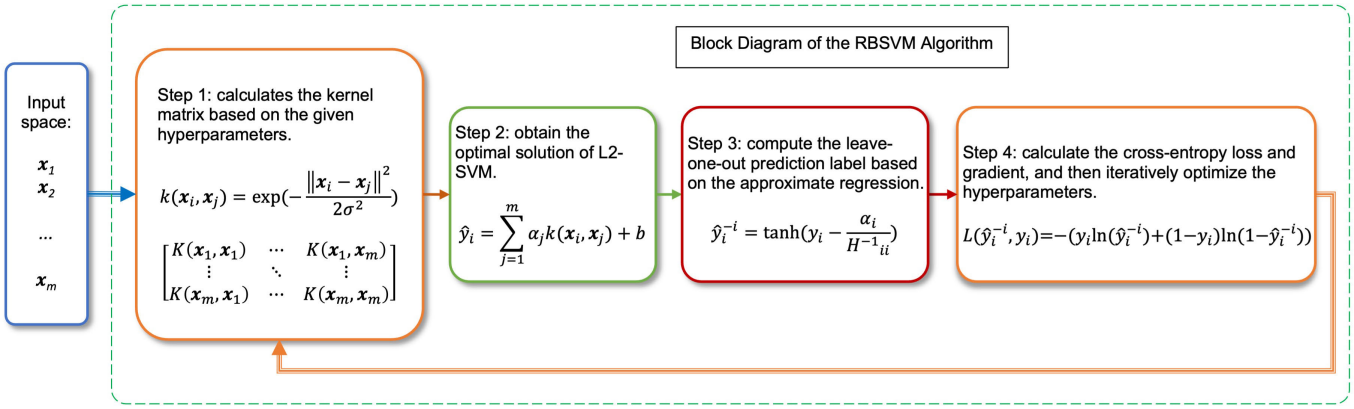

Fig. 3. RBSVM generalization error estimation framework. The tanh function is used to approximate the piecewise regression function equivalent to SVM, and then the leave-one-out method is used to estimate the cross-entropy generalization error. This estimation is used for hyperparameter learning and uncertainty analysis. In the framework, α_i is the optimal solution of L2-SVM, $\hat{y}_i^{-i}$ is the leave-one-out prediction label of sample $\boldsymbol{x_i}$, $L(\hat{y}_i^{-i}, y_i)$ is the leave-one-out cross-entropy loss, and H_{ii}^{-i} is the ith element of the main diagonal of the inverse of the extended kernel matrix.

where m_{11} is the element H_{11}, and M_1 is the block matrix after removing $\boldsymbol{x}_1$. Therefore, the optimal solution after removing $\boldsymbol{x}_1$ is

$$\begin{bmatrix}\boldsymbol{\alpha}^{-1}\\ b^{-1}\end{bmatrix}=M_1^{-1}[y_2,\ldots,y_m,0]^T. \tag{24}$$

Therefore, the prediction for the removed sample $\boldsymbol{x}_1$ is

$$\begin{aligned}\hat{y}_1^{-1}&=\boldsymbol{m}_1^T[\boldsymbol{\alpha}^{-1}\ b^{-1}]^T\\&=\boldsymbol{m}_1^T M_1^{-1}\alpha_1+\boldsymbol{m}_1^T[\alpha_2,\ldots,\alpha_n,b]^T.\end{aligned} \tag{25}$$

On the other hand, according to the KKT optimal condition (23), there are $y_1=m_{11}\alpha_1+\boldsymbol{m}_1^T[\alpha_2,\ldots,\alpha_n,b]^T$. Combining the above Formulation (25), a relationship can be established for $\hat{y}_1^{-1}$ and y_1

$$\hat{y}_1^{-1}=y_1-\alpha_1(m_{11}-\boldsymbol{m}_1^T M_1^{-1}\boldsymbol{m}_1)=y_1-\frac{\alpha_1}{(H^{-1})_{11}} \tag{26}$$

where the last equation is the Woodbury formula $(H^{-1})_{11}=m_{11}-\boldsymbol{m}_1^T M_1^{-1}\boldsymbol{m}_1$ [43]. In the derivation process, since the formula is not sensitive to the sequence number of the sample, the sample sequence can be perturbed to get the same result. Therefore, the superscript 1 can be replaced with i, and the following formula can be obtained [43]:

$$\hat{y}_i^{-i}=y_i\left(1-\frac{\alpha_i}{(H^{-1})_{ii}}\right). \tag{27}$$

In Formulation (27), the leave-one-out prediction value $\hat{y}_i^{-i}$ is discontinuous, and the generalization error estimation based on this is also discontinuous [38]. To calculate its gradient to guide hyperparameter learning, it needs to be converted into a continuous estimation. RBSVM approximates its predicted value to tanh regression output, and then calculates its leave-one-out generalization error estimation by cross-entropy. The approximate tanh output of SVM $\tilde{y}_i^{-i}$ is

$$\tilde{y}_i^{-i}=\tanh(\hat{y}_i^{-i}) \tag{28}$$

where $\tilde{y}_i^{-i}$ and $\hat{y}_i^{-i}$ have the same gradient at the discrimination interface.

RBSVM learns the hyperparameters of the model so that the output of the leave-one-out method approaches the binomial distribution of the real data as much as possible. This method is similar to but different from maximum likelihood estimation. The difference between RBSVM and maximum likelihood estimation is that maximum likelihood estimation is based on the training set for parameter learning, while RBSVM uses leave-one-out prediction for model hyperparameter learning instead of parameter learning.

The range of tanh function is $[-1, 1]$, which can be converted into probability by scaling. Similarly, the sample label value can also be scaled to the binomial distribution, and then the cross-entropy loss can be calculated. The leave-one-out cross-entropy loss is

$$\begin{aligned}&L(y_i,\tilde{y}_i^{-i})\\&=-\left(\frac{1+y_i}{2}\log\frac{1+\tilde{y}_i^{-i}}{2}+\frac{1-y_i}{2}\log\frac{1-\tilde{y}_i^{-i}}{2}\right)\\&=-y_i\hat{y}_i^{-i}+\log\left(\exp(\hat{y}_i^{-i})+\exp(-\hat{y}_i^{-i})\right)\\&=-(1+y_i)\hat{y}_i^{-i}+\log\left(1+\exp(2\hat{y}_i^{-i})\right)\end{aligned} \tag{29}$$

where y_i is the sample label, $\hat{y}_i^{-i}$ is the leave-one-out output of SVM, and $\tilde{y}_i^{-i}$ is the leave-one-out prediction of RBSVM

$$\tilde{y}_i^{-i}=\tanh(\hat{y}_i^{-i})=\tanh\left(y_i-\frac{\alpha_i y_i}{(H^{-1})_{ii}}\right). \tag{30}$$

RBSVM performs hyperparameter learning by minimizing the leave-one-out cross-entropy, so it is necessary to calculate the gradient of the cross-entropy with respect to the hyperparameters. The gradient of the cross-entropy loss function $L(y_i,\tilde{y}_i^{-i})$ with respect to the hyperparameters is as follows:

$$\begin{aligned}&\nabla_\theta L(y_i,\tilde{y}_i^{-i})\\&=-\nabla_\theta\left(\frac{1+y_i}{2}\log\frac{1+\tilde{y}_i^{-i}}{2}+\frac{1-y_i}{2}\log\frac{1-\tilde{y}_i^{-i}}{2}\right)\\&=\nabla_\theta\left(-y_i\hat{y}_i^{-i}+\log\left(\exp(\hat{y}_i^{-i})+\exp(-\hat{y}_i^{-i})\right)\right)\\&=-y_i\nabla_\theta\hat{y}_i^{-i}+\tanh(\hat{y}_i^{-i})\nabla_\theta\hat{y}_i^{-i}\\&=(\tilde{y}_i^{-i}-y_i)\nabla_\theta\hat{y}_i^{-i}\end{aligned} \tag{31}$$

and the gradient of the RBSVM output $\tilde{y}_i^{-i}$ with respect to the hyperparameters is

$$\nabla_\theta \tilde{y}_i^{-i} = \left(1 - \left(\tilde{y}_i^{-i}\right)^2\right) \nabla_\theta \hat{y}_i^{-i}. \tag{32}$$

Then, the objective function of leave-one-out cross-entropy $J(\boldsymbol{\theta})$ is

$$\begin{aligned} J(\theta) &= \sum_{i=1}^{m} L\left(y_i, \tilde{y}_i^{-i}\right) \\ &= -\sum_{i=1}^{m} \left((1+y_i)\hat{y}_i^{-i} - \log\left(1+\exp\left(2\hat{y}_i^{-i}\right)\right)\right) \end{aligned} \tag{33}$$

where $\boldsymbol{\theta}$ is its hyperparameter (such as $\boldsymbol{\theta} = \{\sigma, C\}$ for the Gaussian kernel), and m is the number of samples. The corresponding gradient formulation with respect to its hyperparameters is

$$\nabla_\theta J(\theta) = \sum_{i=1}^{m} \nabla_\theta L\left(y_i, \tilde{y}_i^{-i}\right) = \sum_{i=1}^{m} \left(\tilde{y}_i^{-i} - y_i\right) \nabla_\theta \hat{y}_i^{-i} \tag{34}$$

where $(\tilde{y}_i^{-i} - y_i)\nabla_\theta \hat{y}_i^{-i}$ is the partial derivative of the cross-entropy loss function Formulation (29), $\tilde{y}_i^{-i}$ is the leave-one-out prediction of RBSVM, and y_i is the label of sample i. $\nabla_\theta \hat{y}_i^{-i}$ is the partial derivative of the leave-one-out prediction with respect to the hyperparameter θ, calculated as follows:

$$\begin{aligned} \nabla \hat{y}_i^{-i} &= -y_i \left(\frac{\nabla \alpha_i}{(H^{-1})_{ii}} + \alpha_i \frac{\nabla H_{ii}^{-1}}{(H^{-1})_{ii}^2} \right) \\ &= -y_i \left(\frac{-H_{ii}^{-1} \nabla H_{ii} \alpha_i}{(H^{-1})_{ii}} + \alpha_i \frac{-H_{ii}^{-1} \nabla H_{ii} H_{ii}^{-1}}{(H^{-1})_{ii}^2} \right) \\ &= y_i \alpha_i \nabla H_{ii} \left(1 + \frac{1}{H_{ii}^{-1}}\right) \end{aligned} \tag{35}$$

where $\nabla \alpha_i = -H_{ii}^{-1} \nabla H_{ii} \alpha_i$, and $\nabla H_{ii}^{-1} = -H^{-1} \nabla H_{ii} H_{ii}^{-1}$ [43]. H_{ii}^{-1} is the element ii of H^{-1}, and ∇H_{ii} is the partial derivative of matrix H to the parameter θ.

Substituting Formulation (35) into Formulation (34), we can get

$$\begin{aligned} \nabla J(\theta) &= \sum_{i=1}^{m} \left(\tilde{y}_i^{-i} - y_i\right) \nabla_\theta \hat{y}_i^{-i} \\ &= \sum_{i=1}^{m} \alpha_i y_i \left(\tilde{y}_i^{-i} - y_i\right) \nabla H_{ii} \left(1 + \frac{1}{H_{ii}^{-1}}\right). \end{aligned} \tag{36}$$

In Formulation (36), ∇H_{ii} depends on the specific kernel function. For example, in the Gaussian kernel function, the hyperparameter $\boldsymbol{\theta}$ is $\{\sigma, C\}$. The gradients of the extended kernel matrix H to σ and C are

$$\begin{aligned} \nabla_\sigma H_{ij} &= \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2\sigma^2}\right) \left(\frac{1}{4}||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 \sigma^{-3}\right) \\ &= K_{ij} \left(\frac{1}{4}||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 \sigma^{-3}\right) \\ \nabla_C H_{ii} &= -\frac{1}{C^2} \end{aligned} \tag{37}$$

where K is the kernel matrix. To ensure that the hyperparameters C and σ are nonnegative, let $C = \exp(\hat{c})$ and $\sigma = \exp(\hat{\sigma})$. Therefore, the gradient of H with respect to $\hat{C}$ and $\hat{\sigma}$ is

$$\begin{aligned} \nabla_{\hat{\sigma}} H_{ij} &= K_{ij} \left(\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||}{2\exp(\hat{\sigma})}\right)^2 = -2K_{ii} \log(K_{ii}) \\ \nabla_{\hat{c}} H_{ii} &= -\exp(-\hat{c}). \end{aligned} \tag{38}$$

C. RBSVM Algorithm

RBSVM is a kind of hyperparameter learning algorithm based on generalization error gradient. The leave-one-out cross-entropy generalization error is used to guide hyperparameter learning. The pseudocode of RBSVM is summarized in Algorithm 1.

Algorithm 1 RBSVM

Input: Training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$.
Output: Optimal SVM hyperparameters $\{\sigma, C\}$, and SVM parameters $\{\boldsymbol{\alpha}, b\}$.
Iteration:
1. Initialize $\{\sigma, C\}$, for example $\sigma = 1$, $C = 0$.
2. Obtain the optimal solution of L2-SVM with fixed hyperparameters $\{\sigma, C\}$.
3. Compute $\hat{y}_i^{-i}$ and cross-entropy loss according to (27) and (29).
4. Compute the gradient of cross-entropy with hyperparameter $\{\sigma, C\}$ according to (36) and (37).
5. Update the hyperparameter $\{\sigma, C\}$ to minimize cross-entropy with the gradient descent algorithm.
6. Go back to step 2 or stop when the minimum is reached.

In Algorithm 1, RBSVM can be solved by the fastest gradient descent or conjugate gradient descent method, such as Polack–Ribiere conjugate gradient method [47]. Due to its simplicity and low memory requirements, the conjugate gradient method is one of the most commonly used methods to solve smooth unconstrained optimization problems, among which Polack–Ribiere method is considered to be very effective.

D. Uncertainty Analysis and Model Entropy of RBSVM

Uncertainty can be divided into two major types: epistemic uncertainty and aleatoric uncertainty [48]. Epistemic uncertainty, also known as model uncertainty, is usually the result of a lack of sufficient data. Aleatoric uncertainty comes from the data and is caused by the noise of the data. Therefore, even if more data are obtained, the aleatoric uncertainty does not decrease. For the classification task, Zhu and Wu [16] conducted a systematic evaluation of the impact of noise and summarized some interesting observations. These observations can help researchers design various noise handling mechanisms to improve data quality. They categorized noise into class noise and attribute noise and analyzed their properties and effects separately. For class noise, directly removing samples containing class noise can generally improve the classification accuracy. However, it may not be the best

approach to directly remove samples that contain attribute noise, because the remaining attributes in these samples may contain valuable information. The noise on different attributes has a different impact on model performance. The higher the correlation between an attribute and a class, the greater the negative impact of its noise. Therefore, noise processing mechanisms need to focus more on noise-sensitive attributes rather than treating them equally [16].

This section mainly discusses the uncertainty of SVM and proposes model entropy to describe the uncertainty of the classification model. Although the classification margin of SVM reflects the structural error, the range of the margin exceeds the range of [0, 1] and is related to the penalty coefficient C, which does not directly reflect the uncertainty of the model. Therefore, we propose the concept of model entropy to measure the uncertainty of the classification model. The model entropy is based on the average entropy of the leave-one-out method. The distribution of training data is used to approximate the overall distribution to solve the problem that the classification margin cannot reflect the sample distribution.

The objective of RBSVM is to minimize the cross-entropy loss between the leave-one-out prediction and the sample label. The minimization of cross-entropy loss is equivalent to the minimization of KL divergence, that is, the prediction of RBSVM is used to approximate the true distribution of the sample. Therefore, the product of the output of RBSVM and the sample label are scaled to [0, 1] to form the probability of $\tilde{y} == y$. By calculating the prediction probability of each sample, the model entropy is formed to measure the uncertainty of the model. Its definition is given as follows.

Definition 1 (Model Entropy): Model entropy is used to measure the uncertainty of the model in prediction, and its value range is [0, 1]. The larger the value, the greater the uncertainty of model prediction. It is defined as

$$E_M = -\frac{1}{n}\sum_{i=1}^{n}[P_{y_i}\log(P_{y_i}) + (1-P_{y_i})\log(1-P_{y_i})] \quad (39)$$

where P_{y_i} represents the probability that the prediction of the i sample is correct. For samples with incorrect predictions, the probability value is fixed at $P_{y_i} = 0.5$, and the entropy reaches the maximum value. If the sample prediction is correct, then

$$P_{y_i} = \begin{cases} (1+\tilde{y}_i)/2, & \text{if } y_i = +1 \\ (1-\tilde{y}_i)/2, & \text{if } y_i = -1. \end{cases} \quad (40)$$

The model entropy is different from the classification margin of SVM. The difference is mainly manifested in two aspects: First, the margin of SVM only considers support vectors and does not consider the distribution of nonsupport vectors. Second, choosing different regular hyperparameters will result in different margin, and it is difficult to balance the impact of training errors and the margin on model uncertainty. The model entropy is based on the prediction of the leave-one-out method, not the training error. Therefore, it can better reflect the actual predictive performance of the model.

Although the classification margin and model entropy are different, there is a close relationship between them. Generally, the larger the classification margin, the closer the predicted value $\tilde{y}$ is to the sample label y, and the smaller the model entropy.

TABLE I
EXPERIMENTAL DATA INFORMATION

No.	Dataset	Samples	Features	Classes
1	breast-cancer	286	9	2
2	spect-heart	267	22	2
3	hepatitis	155	19	2
4	diabetis	768	8	2
5	thyroid	215	5	2
6	monks	432	6	3
7	tic-tac-toe	958	9	2
8	adult	1213	14	2
9	crx	690	15	2
10	cmc	1473	9	2
11	twonorm	7400	20	2
12	balance	625	4	3
13	banana	5300	2	2
14	german	1000	20	2
15	ijcnn1	49990	22	2
16	skin-nonskin	10000	3	2

IV. EXPERIMENTS

We analyze and verify the performance of RBSVM from the aspects of hyperparameter learning and model uncertainty. The baseline algorithms include grid search, Bayesian, RM, and span hyperparameter learning.

A. Experimental Datasets and Experimental Settings

The RBSVM algorithm proposed in this article is tested on 16 datasets. The datasets come from the UCI dataset [49] and LibSVM [50]. Table I lists the detailed information of all the experimental datasets used in our experiments. For multiclassification data, we used a one-versus-all method to convert it into binary-classification data. Theory and experiments show that the one-versus-all strategy is very simple and powerful, and its results are often at least as accurate as other methods [41], [51], [52]. For multiclassification data with N classes, the one-versus-all scheme trains N binary classifiers, and each classifier takes one of the class samples as positive and the remaining samples as negative classes. In our experiments, monks data had three class labels {1, 2, 3}. We take one of the classes as positive and the remaining two classes as negative. For example, when the {1} class of monks is positive, the remaining two classes {2, 3} are treated as negative, which is denoted by monks-1. The three-classification monks data are correspondingly decomposed into three two-classification data, represented as monks-1, monks-2, and monks-3, and then three binary classifiers are trained. The same one-versus-all method is applied to another multiclassification dataset balance.

The comparative experiments all use the radial basis kernel function (RBF). The RBF kernel function, also known as the Gaussian kernel function, is one of the most commonly used methods and has many excellent properties. The hyperparameter of RBF is the width parameter σ, which controls the radial range of the function. Another hyperparameter included in the experiments is the regularization coefficient C. The platform used for performing experiments is Dell Precision 7710.

In the experiment, the grid search uses the tenfold cross-validation method to find the optimal hyperparameters.

TABLE II

LIST OF OPTIMAL HYPERPARAMETERS. THERE ARE TWO HYPERPARAMETERS, NAMELY, THE GAUSSIAN KERNEL PARAMETER σ AND THE REGULARIZATION COEFFICIENT C. RBSVM IS THE HYPERPARAMETER LEARNING ALGORITHM PROPOSED IN THIS ARTICLE. SPANSVM IS HYPERPARAMETER LEARNING BASED ON SPAN MARGIN, AND RMSVM IS RM HYPERPARAMETER LEARNING. GRIDSEARCH, RANDOMSEARCH, AND BAYESSEARCH ARE GRID SEARCH, RANDOM SEARCH, AND BAYESIAN HYPERPARAMETER LEARNING, RESPECTIVELY

Dataset	RBSVM		SpnSVM		RMSVM		GridSearch		RandomSearch		BayesSearch	
	σ	C	σ	C	σ	C	σ	C	σ	C	σ	C
1. breast-cancer	3.9005	3.7979	4.3421	3.8711	3.9283	0.92540	0.0039	0.0039	36.238	97.454	102.36	98.832
2. spect-heart	6.0129	3.3595	1.7195	2.3278	6.1977	2.48500	0.0039	0.0039	10.097	37.983	252.79	47.214
3. hepatitis	5.8777	2.9377	8.8703	3.6909	5.6806	1.85831	4.0000	0.5000	31.891	24.408	90.989	186.86
4. diabetis	9.1474	17.545	9.0826	17.127	5.4155	0.51174	32.000	128.00	34.692	230.98	151.81	75.439
5. thyroid	1.9382	13.896	1.5207	3.7117	1.1809	2.74195	2.0000	128.00	2.7062	102.81	2.5300	88.396
6. monks-1	1.8015	85.618	1.5358	20.944	1.9125	0.58766	4.0000	4.0000	11.027	212.15	11.040	186.70
7. monks-2	1.9515	875.72	1.7213	165.74	1.8015	85.6178	2.0000	128.00	3.3074	250.61	244.80	188.96
8. monks-3	2.9620	826.39	2.2453	108.64	1.8061	30.8556	4.0000	4.0000	11.027	212.15	11.040	186.70
9. tic-tac-toe	2.2153	20.188	2.3974	11.308	4.3590	109.523	4.0000	64.000	6.1030	160.35	5.6574	124.27
10. adult	1.7198	2.6242	0.8874	1.7704	5.7342	0.50358	16.000	2.0000	16.125	30.527	129.52	136.89
11. crx	4.0909	0.5177	3.1659	0.4262	3.9232	0.27584	4.0000	0.1250	66.317	116.98	56.840	116.03
12. cmc	0.3481	50.690	0.6290	165.14	0.1752	19.9075	16.000	64.000	12.884	31.855	20.429	192.25
13. twonorm	3.9715	0.4008	2.5280	0.4773	4.6866	0.40087	2.0000	0.0156	86.365	1.0865	183.24	5.0184
14. balance-1	3.1217	839.50	2.7669	249.59	2.8867	8.58761	4.0000	128.00	4.5363	138.87	4.8811	224.11
15. balance-2	2.9079	926.88	2.3344	568.17	2.1747	1.90632	2.0000	128.00	3.3244	190.20	2.4731	181.73
16. balance-3	2.2009	481.05	2.0243	143.80	2.2122	4.17880	2.0000	128.00	2.2593	231.91	2.6509	189.43
17. german	6.0113	1.0409	5.6368	1.3453	6.1475	0.30717	8.0000	0.5000	95.546	19.028	163.85	55.220
18. australian	6.2752	1.6780	6.6151	0.7465	5.0276	0.42501	0.1250	0.1250	3.3906	174.24	0.4186	92.808
19. ijcnn1	7.5612	3.1361	4.1767	9.0165	4.0884	14.1236	4.0000	8.0000	4.3088	30.284	12.124	154.12
20. skin-nonskin	0.3552	2.0729	0.2545	1.3859	0.2972	2.27256	0.2500	2.0000	1.5703	25.468	0.3554	60.249

To find the right hyperparameters, we create a model for each combination of hyperparameters. The search range for hyperparameters $\log(\sigma)$ and $\log(C)$ is $[-8, +8]$ with a step size of 1. Therefore, there are a total of 256 hyperparameter combinations, and 256 models are trained and evaluated accordingly. Random search uses a random method for hyperparameter learning. We focus on the random search, that is, independent draws from a uniform density from the configuration space $[2^{-8}, 2^{8}]$ for σ and C. Bayesian hyperparameter learning uses the Hyperopt toolkit [53], which uses tree Parzen estimator (TPE). Bayesian search uses the same distribution as a random search, i.e., the distribution is uniform on $[2^{-8}, 2^{8}]$. The RM algorithm uses RM for hyperparameter learning [42]. The span algorithm uses the span margin proposed by Chapelle et al. [38] for hyperparameter learning.

The choice of the hyperparameter learning algorithm has great influence on the performance of the model. In the experiment, the grid search and random search used the algorithm in the scikit-learn library [54]. The hyperparameter learning algorithm used in RBSVM, RM, and span is a conjugate gradient descent algorithm implemented by Rasmussen and Nickisch [47], which uses Polack–Ribiere conjugate gradient as the search direction.

B. Hyperparameter Learning and Comparative Analysis

The experiment uses the RBF kernel function, and there are two hyperparameters. One of them is the bandwidth hyperparameter σ in the RBF kernel function, and the other hyperparameter is the regularization penalty coefficient C. The hyperparameters learned by different algorithms on different datasets are shown in Table II.

It can be seen from Table II that the hyperparameters of different algorithms are quite different, even though sometimes the accuracy of the algorithms is very close. The main reason is that the strategies of different learning algorithms are quite different.

From the perspective of prediction accuracy, the nonmodel hyperparameter learning algorithms (grid search, random search, and Bayesian search) perform similarly, and the Bayesian algorithm offers the highest accuracy. Grid search and random search are almost the same. This is mainly because the redundancy of the hyperparameters σ and C is relatively small, and the advantages of the random search method are not significant. It can also be seen that the performance of the traditional grid search hyperparameter selection algorithm is stable.

Compared with nonmodel methods, hyperparameter learning algorithms based on model gradient (RBSVM, SpanSVM, and RMSVM) have better prediction performance. The main reason is that gradient-based algorithms can make full use of model information. In addition, another reason for the low accuracy of grid search is that the granularity and search range of grid search are limited.

It can be seen from Table III that among the hyperparameter learning algorithms, the RBSVM algorithm has the highest accuracy because it estimates the model generalization error more accurately. The span algorithm is better than the RM algorithm. The main reason is that the generalization error estimation of span is more accurate than that of RM. This is consistent with the findings in [38].

To further analyze the difference in prediction accuracy between the models, we use the paired-sample Wilcoxon signed-rank test to perform the statistical significance test. This method is a nonparametric test with no assumptions about the distribution of the sample. The paired-sample Wilcoxon signed-rank test is used to infer whether there is a difference in the median of the distribution of two populations from which the paired samples are drawn. The null hypothesis for this

TABLE III

COMPARISON AND ANALYSIS OF PREDICTION ACCURACY BETWEEN RBSVM AND OTHER ALGORITHMS. IN THE TABLE, THE ACCURACY IS THE AVERAGE VALUE OF TENFOLD CROSS-VALIDATION, THE STANDARD DEVIATION IS IN PARENTHESES, AND THE BOLDFACE IS THE OPTIMAL VALUE

Dataset	RBSVM(%)	SpanSVM(%)	RMSVM(%)	GridSearch(%)	RandomSearch(%)	BayesSearch(%)
1. breast-cancer	**75.51**(0.70)	74.45(0.60)	74.11(0.66)	70.30(1.40)	70.30(1.40)	70.30(7.58)
2. spect-heart	**84.26**(0.18)	83.87(0.96)	83.86(1.22)	79.40(0.17)	79.40(0.17)	79.41(4.82)
3. hepatitis	**89.08**(0.89)	**89.08**(0.89)	**89.08**(0.89)	88.40(06.20)	87.80(0.66)	87.79(6.57)
4. diabetis	**77.60**(0.18)	77.59(0.17)	77.59(0.17)	75.60(0.90)	75.60(0.90)	75.63(9.03)
5. thyroid	**98.12**(0.06)	95.82(0.11)	96.30(0.18)	94.50(5.70)	94.50(0.57)	94.50(5.71)
6. monks-1	**99.78**(0.01)	90.53(0.02)	83.32(0.30)	77.50(17.10)	77.00(17.10)	78.20(17.68)
7. monks-2	**97.21**(0.19)	96.75(0.18)	95.59(0.16)	75.90(12.60)	74.10(10.70)	77.64(16.62)
8. monks-3	**100.0**(0.00)	**100.0**(0.00)	**100.0**(0.00)	**100.0**(0.00)	97.20(0.43)	98.60(0.18)
9. tic-tac-toe	**100.0**(0.00)	99.16(0.01)	98.43(0.01)	96.30(0.58)	97.70(0.50)	97.80(4.99)
10. adult	88.67(0.09)	**88.70**(0.06)	83.40(0.13)	82.90(0.16)	82.90(0.12)	82.85(0.13)
11. crx	**86.67**(0.19)	**86.67**(0.19)	**86.67**(0.19)	85.10(15.50)	84.90(17.20)	84.92(17.25)
12. cmc	77.80(0.10)	77.60(0.10)	77.60(0.10)	72.00(3.50)	72.00(3.50)	**78.60**(3.64)
13. twonorm	**98.40**(0.03)	98.30(0.04)	**98.40**(0.03)	97.90(1.30)	97.90(1.30)	97.80(0.12)
14. balance-1	**99.68**(0.01)	99.20(0.01)	99.03(0.02)	96.80(3.80)	97.10(3.70)	97.60(2.90)
15. balance-2	**94.06**(0.17)	92.61(0.32)	92.15(0.19)	93.90(1.40)	92.50(1.40)	92.25(2.74)
16. balance-3	**99.52**(0.01)	99.04(0.01)	99.04(0.01)	98.90(1.80)	99.00(1.60)	98.71(1.88)
17. german	**76.40**(0.15)	76.00(0.19)	76.00(0.19)	73.30(12.10)	73.40(10.30)	73.30(12.85)
18. australian	**87.54**(0.08)	87.39(0.11)	87.39(0.11)	87.10(0.50)	87.10(0.52)	87.11(5.24)
19. ijcnn1	**97.77**(0.02)	97.37(0.04)	96.27(0.05)	95.80(0.02)	95.60(0.24)	95.55(0.25)
20. skin-nonskin	**99.82**(0.01)	99.70(0.01)	99.52(0.02)	99.60(0.05)	99.50(0.08)	99.60(0.06)
average	**91.39**(0.15)	90.49(0.19)	89.69(0.23)	87.06(4.36)	86.78(3.52)	87.41(4.82)

TABLE IV

PAIRED-SAMPLE WILCOXON SIGNED-RANK TEST FOR THE PREDICTION ACCURACY OF RBSVM WHEN COMPARED WITH OTHER ALGORITHMS, AND THE SIGNIFICANCE LEVEL IS $\alpha = 0.05$. THE TEST RESULTS SHOW THAT THE PREDICTION ACCURACY OF THE RBSVM ALGORITHM IS SIGNIFICANTLY HIGHER THAN THAT OF OTHER LEARNING ALGORITHMS

Paired algorithm	Rank+	Rank-	Z value-	P value
RBSVM-SpanSVM	151	2	-3.528	0.00042
RBSVM-RMSVM	136	0	-3.517	0.00044
RBSVM-GridSearch	190	0	-3.823	0.00013
RBSVM-RandomSearch	210	0	-3.919	0.00009
RBSVM-BayesSearch	206	4	-3.771	0.00016

test is that there is no difference in the median of the two population distributions. The statistical results are shown in Table IV.

In Table IV, Rank+ represents the sum of positive ranks, Rank− represents the sum of negative ranks, and Z value− is calculated based on negative ranks. From the paired-sample Wilcoxon signed-rank test, the prediction accuracy of RBSVM is significantly higher than those of other algorithms at the significance level of 0.05.

C. Analysis of Running Time

To analyze the performance and efficiency (time complexity) of the RBSVM algorithm on large-scale datasets, we compare and analyze the running time of RBSVM algorithms on datasets with different sizes, as shown in Fig. 4. The specific method is to perform random sampling with different sizes on the same dataset, form datasets of different sizes, and then analyze the running time of the algorithm on these datasets. The experimental dataset is skin-nonskin, with random sample sizes increased from 500 to 10 000 with 500 samples at a time. This results in a total of 20 datasets of different sizes. In the comparison, grid search, random search, and Bayesian search have a multiplicative running time when the search scope is increased. Therefore, the search range was divided into 256 (the search range of σ and C is $[-8, +8]$)

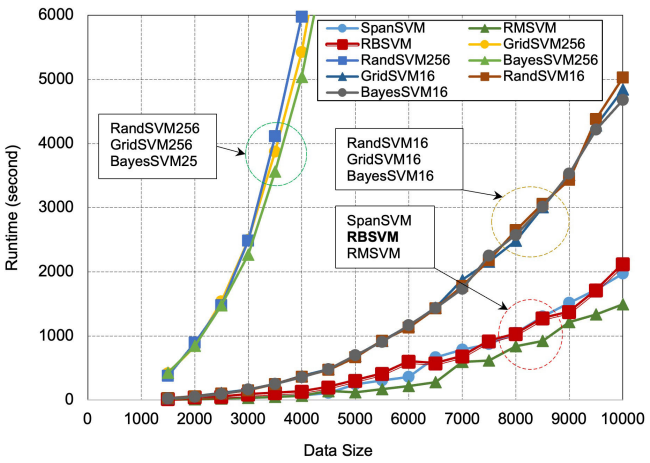

Fig. 4. Run-time comparison between the proposed RBSVM algorithm and other baseline algorithms.

and 16 (the search range of σ and C is $[-2, +2]$). The run time of a group with a search scope of 256 exceeds 5000 s when the data size is greater than 4000. Therefore, the running time of the group with a data size greater than 4000 is not retained in the figure, and only the group with a search size of 16 is retained as a reference for comparison. The comparative experiments were divided into three groups, which were: 1) gradient-based hyperparameter learning algorithms, including SpanSVM, RMSVM, and RBSVM; 2) the group with a search count of 16 (the search range of σ and C is $[-2, 2]$), and its hyperparameter learning algorithm using grid search, random search, and Bayesian search, respectively; and 3) the group with a search count of 256 (the search range of σ and C is $[-8, 8]$), and the hyperparameter learning algorithm also using grid search, random search, and Bayesian search, respectively.

To obtain reliable comparative experiments, all the SVM solving algorithms in the hyperparameter learning algorithms use L2-SVM, and its time complexity is $O(n^3)$. Therefore, as the problem size increases, the run time of all the

TABLE V

ACCORDING TO THE DEFINITION OF MODEL ENTROPY III.2, THE UNCERTAINTY OF MODELS LEARNED BY DIFFERENT ALGORITHMS IS COMPARED. IN THE TABLE, BOLDFACE REPRESENTS THE ALGORITHM WITH THE SMALLEST MODEL ENTROPY, AND ITS UNCERTAINTY IS THE LOWEST

Dataset	RBSVM	SpanSVM	RMSVM	GridSearch	RandomSearch	BayesSearch
1. breast-cancer	0.8643	**0.8631**	0.8682	0.9235	0.8682	0.8801
2. spect-heart	**0.7078**	0.767	0.7608	0.8298	0.7433	0.8093
3. hepatitis	**0.6512**	0.6551	0.6551	0.7244	0.6537	0.6561
4. diabetis	**0.8066**	0.8076	0.8076	0.8093	0.8078	0.8715
5. thyroid	0.391	0.5516	0.4908	**0.1647**	0.2575	0.2516
6. monks-1	**0.5256**	0.6584	0.8482	0.7886	0.7661	0.7749
7. monks-2	**0.4411**	0.4797	0.4847	0.4847	0.5617	0.5072
8. monks-3	**0.1817**	0.2450	0.2781	0.2786	0.4474	0.4887
9. tic-tac-toe	**0.4432**	0.6221	0.5831	0.5402	0.5876	0.6128
10. adult	**0.6162**	0.6168	0.6901	0.7168	0.6405	0.7116
11. crx	**0.8048**	0.8066	0.8066	0.8066	0.7218	0.7635
12. cmc	0.6912	**0.6901**	0.6912	0.9045	0.9028	0.6903
13. twonorm	0.8430	0.8816	**0.8426**	0.9395	0.9359	0.9337
14. balance-1	**0.0888**	0.1091	0.1259	0.1485	0.1578	0.1463
15. balance-2	**0.2359**	0.2906	0.6529	0.2916	0.3621	0.2619
16. balance-3	**0.0952**	0.1132	0.1132	0.1132	0.1000	0.1067
17. german	**0.8159**	0.8289	0.8289	0.8419	0.8662	0.8665
18. australian	**0.6996**	0.7634	0.7634	0.7634	0.7166	0.7165
19. ijcnn1	**0.3935**	0.4268	0.6400	0.6356	0.6457	0.6704
20. skin-nonskin	**0.4431**	0.5595	0.5302	0.6531	0.6624	0.5731
average	**0.5372**	0.5876	0.6050	0.6179	0.6204	0.6147

TABLE VI

SIGNIFICANCE TESTS FOR RBSVM IN MODEL ENTROPIES USING THE PAIRED-SAMPLE WILCOXON SIGNED-RANK TEST AT THE SIGNIFICANCE LEVEL OF 0.05. THE RESULTS SHOW THAT THE MODEL ENTROPY OF RBSVM IS SIGNIFICANTLY LOWER THAN THAT OF OTHER ALGORITHMS

Paired algorithm	Rank+	Rank-	Z value+	P value
RBSVM-SpanSVM	7	203	-3.659	0.00025
RBSVM-RMSVM	1	189	-3.783	0.00016
RBSVM-GridSearch	18	192	-3.248	0.00116
RBSVM-RandomSearch	24	186	-3.024	0.00249
RBSVM-BayesSearch	24	186	-3.024	0.00249

hyperparameter learning algorithms increases rapidly. Among them, the gradient-based hyperparameter learning algorithm can quickly converge to the optimal solution, its iteration number is small, and the number of calls to L2-SVM is also small. Therefore, in practical applications, the gradient-based algorithms often run less number of iterations than grid search, random search, and Bayesian search. This observation is consistent with the literature [37]. The run time of the latter three algorithms depends on the search scope. For a large search range, the run time can be increased to multiple times. For example, in this experiment, the running time of candidate parameters of 256 (16×16, σ and C with a search range of $[-8, 8]$) was almost $16\times$ that of candidate parameters of 16 (4×4). However, if the search scope is small, we may not be able to find suitable hyperparameters. In gradient-based hyperparameter optimization algorithms, the running times of RMSVM, SpanSVM, and RBSVM are comparable with no significant differences. Its running time is mainly related to the number of iterations. Different algorithms have different optimal solutions for convergence, and the number of iterations is also different. In general, if the algorithm converges to a small generalization error, the smaller the test error of its model, the more iterations it has. Generally, RBSVM and SpanSVM converge to better hyperparameter solutions than RMSVM, with slightly more iterations.

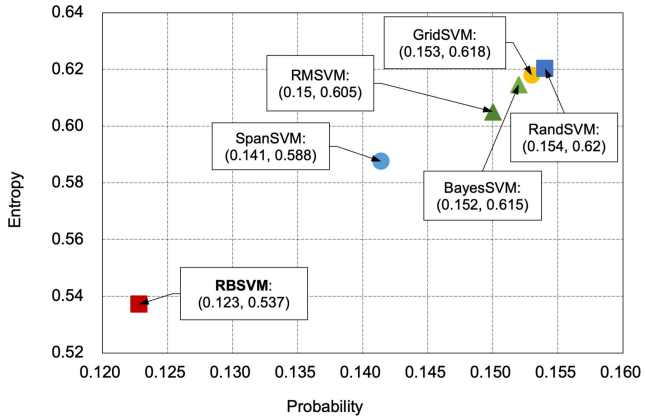

Fig. 5. Comparative analysis of model entropy. The parentheses are (probability, entropy), where the first is the prediction probability, and the second is the model entropy. The horizontal coordinate is the probability of correct prediction, and the vertical coordinate is the corresponding model entropy. In the figure, RBSVM has the lowest model entropy and uncertainty.

D. Uncertainty Analysis of RBSVM

Uncertainty analysis is an important part of the analysis of machine learning algorithms. This section uses the model entropy proposed in this article to analyze the uncertainty of the model learned by different algorithms.

It can be seen from Table V that the uncertainty of the model learned by different hyperparameter learning algorithms is quite different. Therefore, we perform a paired-sample Wilcoxon signed-rank test on the model entropy, and the results are shown in Table VI. In this table, the meanings of Rank+, Rank− and P value are the same as in Table IV. Z value+ is calculated based on positive ranks. It can be seen that the uncertainty of the model of the RBSVM algorithm is the lowest at the significance level of 0.05.

To more intuitively reflect the uncertainty of the model obtained by each hyperparameter learning algorithm, we display the model entropy and the leave-one-out prediction probability value on a 2-D graph, as shown in Fig. 5.

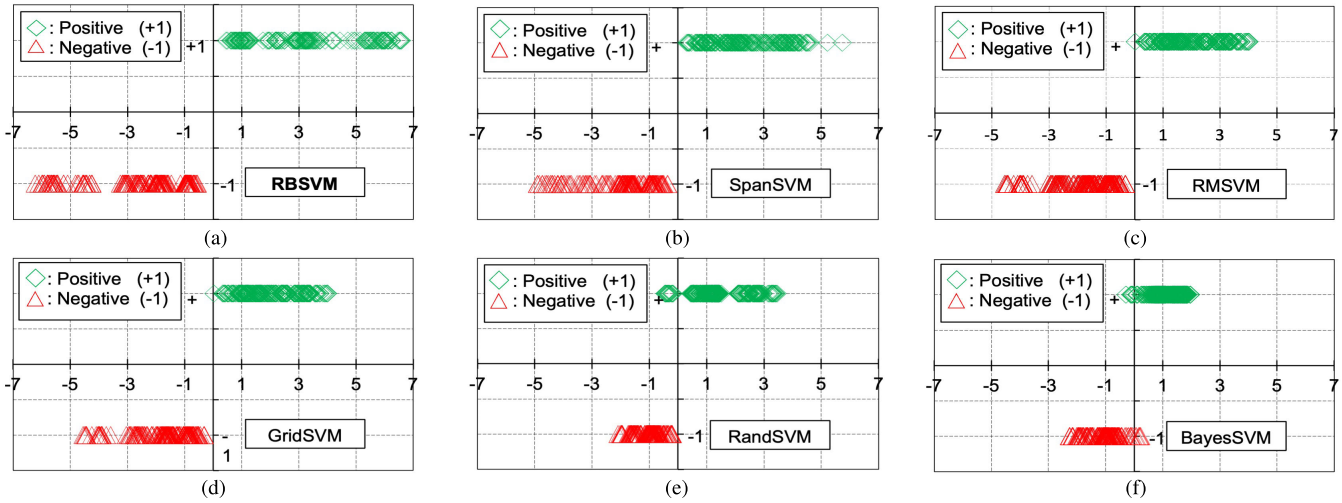

Fig. 6. Prediction distribution of each learning algorithm on the monk dataset. The horizontal coordinate is the cross-validation prediction value of the sample, and the vertical coordinate is the classification label. The prediction accuracy of RBSVM, SpanSVM, RMSVM, and GridSVM is all 100%, but their model entropy is different. The prediction distribution of RBSVM is farthest from the origin. RMSVM and GridSVM have one sample whose prediction is very close to the origin, which is easy to misclassify. RandSVM and BayesSVM have some prediction errors, that is, the prediction of positive sample is less than 0, and the prediction of negative sample is greater than 0. The closer the prediction of the sample is to the boundary (that is, the origin in this figure), the easier it is to predict errors and the smaller the model entropy. (a) RBSVM accuracy: 100%, entropy: 0.182. (b) SpanSVM accuracy: 100%, entropy: 0.245. (c) RMSVM accuracy: 100%, entropy: 0.278. (d) GridSVM accuracy: 100%, entropy: 0.279. (e) RandSVM accuracy: 97.1%, entropy: 0.447. (f) BayesSVM accuracy: 98.6%, entropy: 0.489.

The horizontal coordinate is the probability value predicted by the leave-one-out method, and the vertical coordinate is its corresponding model entropy. It can be seen from the figure that the model entropy of RBSVM is the smallest. The entropies of the grid search, random search, and Bayesian search algorithms are very close. It should be noted that the probability here is the uncertainty probability corresponding to the model entropy, not the prediction error.

Model entropy is different from training or prediction error. Even if the training or prediction error is 0, model entropy can be used to measure the uncertainty of model. To illustrate this situation, we use the data monk as an example. The prediction accuracy of RBSVM, SpanSVM, RMSVM, and grid search is all 100%, but the uncertainty is different. In Fig. 6, the vertical coordinate is the classification label. The horizontal coordinate is the prediction value of the cross-validation. If the classification is correct and the prediction value is farther from the origin of the coordinate, the uncertainty is lower. Although the predictions of RBSVM, SpanSVM, RMSVM, and grid search are all correct, their classification margin and model entropy are quite different. This also explains why the RBSVM algorithm has the highest prediction accuracy in the comparative experiment.

E. Discussion of Kernel Functions and Hyperparameters

Common kernel functions include linear kernel, polynomial kernel, and Gaussian kernel functions. The linear kernel function does not require feature mapping, but instead calculates the inner product in the original sample space and is then used to build a linear model. In contrast, the polynomial kernel allows the learning of nonlinear models by representing the similarity in a feature space. The degree-d polynomial kernel function is defined as

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\boldsymbol{x}_i^T \boldsymbol{x}_j + c\right)^d \tag{41}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are samples in the input space, and c is a constant trading off the influence of higher order versus lower order terms in the polynomial. The hyperparameter c is a nonnegative real number whose value is continuous, and the hyperparameter d is a positive integer whose value is discrete. The Gaussian kernel function is widely used to map samples to infinite-dimensional feature spaces on which the models are built. Its hyperparameter is the bandwidth parameter σ, and its value is continuous. In addition, there are many other kernel functions, such as the Laplacian kernel, Spline kernel, and the Wavelet kernel, to meet the needs of different tasks.

In addition to a single basic kernel function, a new kernel function can be constructed from a combination of basic kernel functions. Among them, the Hermite orthogonal polynomial kernel function proposed by Moghaddam and Hamidzadeh [55] is an interesting example. The Hermite kernel function is a combination of Hermite orthogonal polynomials, which are defined as follows:

$$He_n = (-1)^n \exp\left(\frac{x^2}{2}\right)\frac{d^n}{dx^n}\exp\left(-\frac{x^2}{2}\right) \tag{42}$$

where $He_n(x)$ is an nth-order polynomial for $n = 0, 1, 2, 3, \ldots$. These polynomials are orthogonal to each other. By combining the Hermite polynomials, the Hermite kernel function is constructed as follows:

$$k(x_i, x_j) = \sum_{n=0}^{m} He_n(x_i) He_n(x_j) \tag{43}$$

where m is the number of polynomials used in the combination, and $He_n(x_i)$ and $He_n(x_j)$ represent the Hermite polynomials of order n. The Hermite kernel functions can improve classification accuracy, reduce the number of support vectors, and increase the speed of SVMs.

In different kernel functions, the number and properties of their hyperparameters are often different. Hyperparameters can

be divided into two categories according to their properties. One type of hyperparameters is continuous, such as the bandwidth hyperparameter σ in a Gaussian kernel function. The other type of hyperparameter is discrete, such as order d in a polynomial kernel function. Hyperparameters of different properties are often learned in different ways. This article focuses on the learning of the first type of hyperparameters, such as the hyperparameter σ in the Gaussian kernel function and the regularization hyperparameter C. These hyperparameters can be learned using gradient-based algorithms. For the second type of hyperparameters, such as m in the Hermite kernel function and d in the polynomial kernel functions, it is often not straightforward to use the gradient descent algorithms. Its optimization algorithm can refer to integer optimization methods or other methods [56].

V. Conclusion

In this article, we have presented a new hyperparameter learning algorithm RBSVM based on a novel idea where the maximum margin classification problem is transformed into a regression problem. RBSVM finds the hyperparameters of the maximum margin classification model by optimizing the objective function defined on the unbiased generalization error with a gradient descent algorithm. In addition, we have proposed a new method for model uncertainty measurement based on leave-one-out prediction probability.

The experimental analysis and significance test show that the proposed RBSVM offers significantly higher prediction accuracy than the baseline algorithms, with lower model uncertainty. The training time of the RBSVM algorithm is similar to that of SpanSVM and RMSVM, but less than that of grid search and Bayesian methods.

The proposed RBSVM algorithm is based on a gradient descent method, which can only handle continuous hyperparameters, but not discrete hyperparameters. In the future, we intend to use integer optimization algorithms to deal with the problem of learning with discrete hyperparameters.

Acknowledgment

The authors would like to thank the editors and reviewers for their insightful comments and suggestions that lead to improved quality of this article.

References

[1] A. Tsvieli and N. Weinberger, "Learning maximum margin channel decoders," *IEEE Trans. Inf. Theory*, vol. 69, no. 6, pp. 3597–3626, Jun. 2023.

[2] C. Tzelepis, V. Mezaris, and I. Patras, "Linear maximum margin classifier for learning from uncertain data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2948–2962, Dec. 2018.

[3] S. Abidi, M. Piccardi, I. W. Tsang, and M.-A. Williams, "Well-M^{3}N: A maximum-margin approach to unsupervised structured prediction," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 6, pp. 427–439, Dec. 2019.

[4] B.-B. Jia and M.-L. Zhang, "Maximum margin multi-dimensional classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7185–7198, Dec. 2022.

[5] F. Pernkopf, M. Wohlmayr, and S. Tschiatschek, "Maximum margin Bayesian network classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 521–532, Mar. 2012.

[6] K. H. Kim and S. Y. Sohn, "Hybrid neural network with cost-sensitive support vector machine for class-imbalanced multimodal data," *Neural Netw.*, vol. 130, pp. 176–184, Oct. 2020.

[7] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer, 2013, doi: 10.1007/978-1-4757-2440-0.

[8] M. Avolio and A. Fuduli, "A semiproximal support vector machine approach for binary multiple instance learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3566–3577, Aug. 2021.

[9] Y. Chen, Q. Mao, B. Wang, P. Duan, B. Zhang, and Z. Hong, "Privacy-preserving multi-class support vector machine model on medical diagnosis," *IEEE J. Biomed. Health Informat.*, vol. 26, no. 7, pp. 3342–3353, Jul. 2022.

[10] G. Hoxha and F. Melgani, "A novel SVM-based decoder for remote sensing image captioning," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5404514.

[11] A. Glowacz, "Thermographic fault diagnosis of shaft of BLDC motor," *Sensors*, vol. 22, no. 21, pp. 1–13, 2022.

[12] A. Glowacz, "Thermographic fault diagnosis of electrical faults of commutator and induction motors," *Eng. Appl. Artif. Intell.*, vol. 121, no. 5, pp. 1–15, 2023.

[13] M. Tanveer, T. Rajani, R. Rastogi, Y. H. Shao, and M. A. Ganaie, "Comprehensive review on twin support vector machines," *Ann. Oper. Res.*, pp. 1–46, Mar. 2022, doi: 10.1007/s10479-022-04575-w.

[14] Jayadeva, R. Khemchandani, and S. Chandra, "Twin support vector machines for pattern classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 905–910, May 2007.

[15] M. A. Ganaie, M. Tanveer, and C.-T. Lin, "Large-scale fuzzy least squares twin SVMs for class imbalance learning," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 11, pp. 4815–4827, Nov. 2022.

[16] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artif. Intell. Rev.*, vol. 22, no. 3, pp. 177–210, Nov. 2004.

[17] H. Liu, Y.-S. Ong, Z. Yu, J. Cai, and X. Shen, "Scalable Gaussian process classification with additive noise for non-Gaussian likelihoods," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 5842–5854, Jul. 2022.

[18] M. Tanveer, M. A. Ganaie, A. Bhattacharjee, and C. T. Lin, "Intuitionistic fuzzy weighted least squares twin SVMs," *IEEE Trans. Cybern.*, vol. 53, no. 7, pp. 4400–4409, Jun. 2023.

[19] S. Moslemnejad and J. Hamidzadeh, "A hybrid method for increasing the speed of SVM training using belief function theory and boundary region," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 12, pp. 3557–3574, Dec. 2019.

[20] J. Hamidzadeh and S. Moslemnejad, "Identification of uncertainty and decision boundary for SVM classification training using belief function," *Appl. Intell.*, vol. 49, no. 6, pp. 2030–2045, Jun. 2019.

[21] S. Moslemnejad and J. Hamidzadeh, "Weighted support vector machine using fuzzy rough set theory," *Soft Comput.*, vol. 25, no. 13, pp. 8461–8481, Jul. 2021.

[22] Z. Liang and L. Zhang, "Uncertainty-aware twin support vector machines," *Pattern Recognit.*, vol. 129, Sep. 2022, Art. no. 108706, doi: 10.1016/j.patcog.2022.108706.

[23] K. Nguyen, T. Le, T. D. Nguyen, G. I. Webb, and D. Phung, "Robust variational learning for multiclass kernel models with stein refinement," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4425–4438, Sep. 2022.

[24] C. E. D. S. Santos, R. C. Sampaio, L. D. S. Coelho, G. A. Bestard, and C. H. Llanos, "Multi-objective adaptive differential evolution for SVM/SVR hyperparameters selection," *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107649, doi: 10.1016/j.patcog.2020.107649.

[25] J.-Y. Hsia and C.-J. Lin, "Parameter selection for linear support vector regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5639–5644, Dec. 2020.

[26] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106622, doi: 10.1016/j.knosys.2020.106622.

[27] S. Jain and R. Rastogi, "Parametric non-parallel support vector machines for pattern classification," *Mach. Learn.*, pp. 1–28, Oct. 2022, doi: 10.1007/s10994-022-06238-0.

[28] S. Czaplak and A. Horzyk, "Automatic optimization of hyperparameters using associative self-adapting structures," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8, doi: 10.1109/IJCNN55064.2022.9892758.

[29] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5149–5169, Sep. 2022.

[30] W. E. I. Jiang and S. Siddiqui, "Hyper-parameter optimization for support vector machines using stochastic gradient descent and dual coordinate descent," *EURO J. Comput. Optim.*, vol. 8, no. 1, pp. 85–101, Mar. 2020.
[31] S. Nematzadeh, F. Kiani, M. Torkamanian-Afshar, and N. Aydin, "Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases," *Comput. Biol. Chem.*, vol. 97, Apr. 2022, Art. no. 107619, doi: 10.1016/j.compbiolchem.2021.107619.
[32] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-05318-5.
[33] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
[34] Z. Yang and A. Zhang, "Hyperparameter optimization via sequential uniform designs," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 6592–6638, 2021.
[35] K. J. Prabuchandran, S. Penubothula, C. Kamanchi, and S. Bhatnagar, "Novel first order Bayesian optimization with an application to reinforcement learning," *Appl. Intell.*, vol. 51, no. 3, pp. 1565–1579, Mar. 2021.
[36] M. Lindauer et al., "SMAC3: A versatile Bayesian optimization package for hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 23, no. 1, pp. 2475–2483, 2022.
[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[38] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Mach. Learn.*, vol. 46, pp. 131–159, Jan. 2002.
[39] J. Wainer and G. Cawley, "Empirical evaluation of resampling procedures for optimising SVM hyperparameters," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 475–509, 2017.
[40] S. Wang, Q. Liu, E. Zhu, F. Porikli, and J. Yin, "Hyperparameter selection of one-class support vector machine by self-adaptive data shifting," *Pattern Recognit.*, vol. 74, pp. 198–211, Feb. 2018.
[41] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, Dec. 2004.
[42] S. S. Keerthi, V. Sindhwani, and O. Chapelle, "An efficient method for gradient-based adaptation of hyperparameters in SVM models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 673–680.
[43] G. C. Cawley and N. L. C. Talbot, "Fast exact leave-one-out cross-validation of sparse least-squares support vector machines," *Neural Netw.*, vol. 17, no. 10, pp. 1467–1475, Dec. 2004.
[44] G. C. Cawley and N. L. C. Talbot, "Efficient approximate leave-one-out cross-validation for kernel logistic regression," *Mach. Learn.*, vol. 71, nos. 2–3, pp. 243–264, Jun. 2008.
[45] M. Opper and O. Winther, "Gaussian processes for classification: Mean-field algorithms," *Neural Comput.*, vol. 12, no. 11, pp. 2655–2684, Nov. 2000.
[46] J. Liu, Z. Wu, L. Xiao, and H. Yan, "Learning multiple parameters for kernel collaborative representation classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5068–5078, Dec. 2020.
[47] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (GPML) toolbox," *J. Mach. Learn. Res.*, vol. 11, pp. 3011–3015, Nov. 2010.
[48] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Mach. Learn.*, vol. 110, no. 3, pp. 457–506, Mar. 2021.
[49] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[50] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.
[51] L. Ding and Y. Chen, "Leave-one-out approach for matrix completion: Primal and dual analysis," *IEEE Trans. Inf. Theory*, vol. 66, no. 11, pp. 7274–7301, Nov. 2020.
[52] Y. Zhu, C. Markos, R. Zhao, Y. Zheng, and J. J. Q. Yu, "FedOVA: One-vs-all training method for federated learning with non-IID data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–7, doi: 10.1109/IJCNN52387.2021.9533409.
[53] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proc. Int. Conf. Mach. Learn.*, vol. 28, 2013, pp. 115–123.
[54] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[55] V. H. Moghaddam and J. Hamidzadeh, "New Hermite orthogonal polynomial kernel and combined kernels in support vector machine classifier," *Pattern Recognit.*, vol. 60, pp. 921–935, Dec. 2016.
[56] X. Zhang, M. Fan, D. Wang, P. Zhou, and D. Tao, "Top-k feature selection framework using robust 0–1 integer programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3005–3019, Jul. 2021.

Shili Peng received the M.S. degree from Central South University, Changsha, China, in 2005, and the Ph.D. degree from Tianjin University, Tianjin, China, in 2017.

He has been working with the Guangdong University of Finance, Guangzhou, China, since 2005, where he is currently the Director of the Department of Internet Finance. He has served as a Cadre of the Economic and Information Commission of the Guangdong Provincial Government in 2013. His current research interests include optimization theory, machine learning, and high-dimensional statistics.

Wenwu Wang (Senior Member, IEEE) is currently a Professor of signal processing and machine learning and the Co-Director of the Machine Audition Laboratory, Centre for Vision Speech and Signal Processing, University of Surrey, Guildford, U.K. He is also an AI Fellow with the Surrey Institute for People-Centred Artificial Intelligence, University of Surrey. His current research interests include signal processing, machine learning and perception, artificial intelligence, machine audition (listening). He has more than 350 publications in these areas.

Yinli Chen received the B.Eng. degree from Nankai University, Tianjin, China, in 1985, and the M.E. degree from the Institute of Computer Technology, Chinese Academy of Sciences, Beijing, China, in 1988.

He has worked with the Huazhong University of Science and Technology, Wuhan, China, from 1988 to 1992. In 1992, he joined the Guangdong University of Finance, Guangzhou, China, where he is currently a Full Professor and the Head of the Department of Computer Science and Technology. His research interests include pattern analysis and machine learning in financial markets.

Xueling Zhong received the Ph.D. degree in business administration from the School of Management, Jinan University, Guangzhou, China, in 2011.

He is currently a Professor with the School of Internet Finance and Information Engineering, Guangdong University of Finance, Guangzhou. His current research interests include production scheduling and neural combinatorial optimization.

Qinghua Hu (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 1999, 2002, and 2008, respectively.

He was a Post-Doctoral Fellow with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, from 2009 to 2011. He is currently the Chair Professor of the College of Intelligence and Computing, Tianjin University, Tianjin, China; and the Director of the SIG Granular Computing and Knowledge Discovery and the Chinese Association of Artificial Intelligence. He is supported by the Key Program, National Natural Science Foundation of China. He has published more than 300 peer-reviewed articles. His current research interests include uncertainty modeling in big data, machine learning with multimodality data, and intelligent unmanned systems.