IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 31, 2020, accepted June 30, 2020, date of publication July 6, 2020, date of current version July 29, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3007499*

# Parallel Machine Learning Algorithm Using Fine-Grained-Mode Spark on a Mesos Big Data Cloud Computing Software Framework for Mobile Robotic Intelligent Fault Recognition

**GUANGMING XIAN, (Member, IEEE)**
School of Software, South China Normal University, Foshan 528225, China
e-mail: xgm20011@sina.com

This work was supported in part by the National Natural Science Foundation of China, a Computing Model Based on Formal Domain Fusion, under Grant 61070015, in part by the South China Normal University, and in part by the South China University of Technology.

**ABSTRACT** An accurate and efficient intelligent fault diagnosis of mobile robotic roller bearings can significantly enhance the reliability and safety of mechanical systems. To improve the efficiency of intelligent fault classification of mobile robotic roller bearings, this paper proposes a parallel machine learning algorithm using fine-grained-mode Spark on a Mesos big data cloud computing software framework. Through the segmentation of datasets and the support of a parallel framework, the parallel processing technology Spark is combined with a support vector machine (SVM), and a parallel single-SVM algorithm is realized using Scala language. In this approach, empirical mode decomposition (EMD) is applied to extract the energy of the acceleration vibration signal in different frequency bands as features. The parallel EMD-SVM approach is applied to detect faults in mobile robotic roller bearings from fault vibration signals. The experimental results show that it can accurately and effectively identify the faults, and it outperforms existing methods based on parallel deep belief network (DBN) and parallel radial basis function neural network under different training set sizes. Fault classification tests are conducted on outer-race and inner-race faults: in both cases, the proposed parallel EMD-SVM outperforms the serial EMD-SVM in terms of the classification accuracy and classification time under different test sizes. For a small number of nodes, the processing time of the proposed Spark model is less than that of Hadoop but close to that of Storm. For 17 slave nodes, the average precision, average recall, and average *F1* score of Spark on Mesos in the fine-grained mode reach 97.3, 97.8, and 97.9%, respectively. The parallel EMD-SVM algorithm in the big data Spark cloud computing framework can improve the accuracy of intelligent fault classification, albeit by a small margin, with higher processing speed and learning convergence rate.

**INDEX TERMS** Parallel machine learning algorithm, parallel support vector machine, mesos cluster manager, big data Spark, cloud computing software framework, empirical mode decomposition, intelligent fault recognition, mobile robotic roller bearing, parallel deep belief network.

## I. INTRODUCTION

Big data processing technologies are being rapidly developed given the increase in the amount of information being stored in recent years. Apache Spark, a parallel computing software framework developed on the basis of Hadoop, is ideal for large-scale iterative computing in a cloud computing software framework [1], [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Xiao Liu.

The intelligent fault diagnosis of mobile robotic roller bearings involves obtaining information, extracting features, and identifying conditions. More specifically, the features of fault vibration signals collected from a mobile robotic roller bearing are extracted by empirical mode decomposition (EMD) [3], [4] and then inputted to a parallel support vector machine (SVM) [5], [6] based on a big data cloud computing software framework for classification. Considering the non-stationary characteristics of mobile robotic roller bearing fault vibration signals, this paper proposes a mobile robotic

roller bearing fault diagnosis approach based on EMD and parallel SVM.

Huang *et al.* recently proposed an improved EMD time-frequency analysis method. Because feature extraction through EMD is purely based on the properties extracted from the data without considering the concept of stationarity, it was proposed for analyzing nonlinear and nonstationary signals, unlike other analysis techniques such as Fourier transforms and wavelet decomposition. The EMD technique can decompose any complicated dataset into a finite and often small number of intrinsic mode functions (IMFs). The IMF is complete, adaptive, and largely orthogonal. As IMFs can be treated as mono-components, the instantaneous frequency of a nonlinear and nonstationary signal can be determined through the Hilbert transformation [7]–[9]. The EMD method has been employed in applications ranging from rainfall analysis [9] to fault diagnosis of robotic roller bearings [10]–[12].

The SVM has been receiving increasing attention in areas ranging from its original application in pattern recognition [13]–[15] to fields such as fault detection of mechanical components. Corinna and Vapnik [16] proposed the current version of the SVM in 1995. The SVM [17] is based on structural risk minimization (SRM), which is different from the commonly used empirical risk minimization (ERM) principle. Owing to the SRM principle, the SVM typically exhibits a higher generalization performance than traditional neural network methods. Training the SVM is equivalent to solving a linearly constrained quadratic programming problem. A disadvantage, however, is the training time scale, which lies somewhere between quadratic and cubic scales with respect to the number of training samples [18].

In view of the problems of slow optimization and high memory consumption in training the SVM algorithm with large-scale data, a parallel SVM algorithm based on the Spark platform is proposed. The MapReduce computing model needs to access the disk many times during iterative processing, which affects the training speed. Spark, which is a distributed cluster framework based on memory, has advantages such as scalability, reliability, and load balancing. It is a parallel computing model that can efficiently process big data. In this study, parallel SVM and Spark computing platform are thoroughly analyzed. Based on data partition, an efficient parallel SVM algorithm is implemented. The experimental results show that the algorithm can accelerate the convergence of the model, improve the classification efficiency, and is suitable for large-scale data processing.

The proposed parallel EMD-SVM [19]–[23] machine learning method based on Spark is utilized for the fault diagnosis of robotic roller bearings. First, the original acceleration vibration signal is decomposed by EMD, and eight IMF components are obtained. The theory of EMD energy entropy is introduced, which can reflect the real working condition and the fault type of robotic roller bearing. The kurtosis value of each IMF component of a robotic roller bearing signal describes the vibration distribution features of the component signal and the strength of the fault impact signal in each order component. For a better fault diagnosis of mobile robotic roller bearings in their working condition, a parallel SVM using fine-grained-mode Spark on Mesos is developed as an identifier, and the extracted energy features of the stationary IMFs are taken as network input vectors. Thus, we could classify faulty and non-faulty bearings [10].

The parallel SVM based on Spark has thus far not been applied to large-scale fault classification and recognition. This study combines Spark with an SVM to realize a parallel SVM approach that can improve the recognition efficiency of fault patterns and thereby the classification accuracy. This study can serve as a theoretical basis for the automatic and rapid diagnosis of fault patterns, and has practical significance for promoting the combined application of parallel machine learning, signal processing, and big data and cloud computing technologies to mobile robotic intelligent fault diagnosis.

The following are the main contributions of this paper:

(1) A new approach using the EMD is proposed for extracting the features from the vibration signals of mobile robotic roller bearings and a parallel SVM based on Spark to classify the different types of faults from the features extracted through the EMD of different fault patterns.

(2) The average recognition times of different computing platforms under different cluster and input data sizes are evaluated. The experimental results show that the proposed parallel machine learning based on the Spark model exhibits a lower average recognition time than Storm and Hadoop.

(3) The average recognition performance is evaluated in different distributed deployment modes of Spark under different cluster and input data sizes. In terms of the recognition time and accuracy, Spark on Mesos in the fine-grained mode requires a slightly longer recognition time than Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method. Therefore, Spark on Mesos in the fine-grained mode is chosen as the distributed deployment mode.

The rest of this paper is organized as follows. Section II presents the EMD feature extraction basic theory, where the energy features extracted from eight IMFs are used as input vectors to the parallel SVM. Section III outlines the SVM machine learning method fault diagnosis process. Section IV introduces the parallel EMD-SVM based on Spark on Mesos in a cloud computing network. Section V presents the experimental results and discussion. To evaluate the average recognition accuracy of the proposed parallel EMD-SVM algorithm, experiments are performed, and the results are compared with those of state-of-art algorithms. The classification results of parallel EMD-SVM, parallel EMD-DBN, and parallel EMD-RBFNN in the cloud computing framework are compared under different training set sizes. To test the performance of the proposed machine learning technique in noisy environments, test sets of noises with signal-to-noise ratio (SNR) values ranging from 15 to 60 dB are considered. The average recognition accuracy for fault identification with Spark on Mesos in the fine-grained

mode under different cluster and input data sizes is found to be better than those of Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method. Finally, the conclusions of this study are given in Section VI.

## II. BASIC THEORY OF THE EMPIRICAL MODE DECOMPOSITION FEATURE EXTRACTION METHOD

### A. INTRINSIC MODE FUNCTION

Huang *et al.* proposed the IMF. The IMF is a function that meets the following two conditions [24], [25]:

(1) In the entire signal interval, the number of extreme points and zero points are equal or differ by 1.

(2) At any point in the signal interval, the mean value of the upper and lower envelopes formed by the local maximum and minimum is zero.

The first condition, which is similar to the requirement of narrowband signals, is of global significance. The second condition is the local condition of the signal, giving it a local significance.

The IMF essentially reflects the inherent volatility of the signals. Under the above two conditions, in each period of the signal, there is a unique wave shape, and there is no multi-modal superposition, so the signal has a single frequency at a certain time. Therefore, we can use the IMF to solve its instantaneous frequency. Huang *et al.* proved that the instantaneous frequency obtained by Hilbert transformation of the IMF is consistent with previous research and the physical mechanism of the system.

### B. EMPIRICAL MODE DECOMPOSITION

In practice, most of the signals are multi-modal mixed signals, including more than one vibration mode. Not all of them meet the IMF conditions. Based on the above theory, Huang *et al.* proposed the EMD method. The basic principle of the EMD is as follows: For a real signal, the Hilbert transform cannot be used to describe the frequency component directly, so it is necessary to first apply the EMD to divide the original signal into several IMF components, and then carry out the Hilbert transform on each component to obtain their instantaneous frequencies [24], [25].

The EMD method can be applied to any type of signal decomposition in theory, making it more advantageous than the previous method in dealing with non-stationary and nonlinear data. Therefore, the EMD method has been widely and effectively applied in various engineering fields since its proposal. In the EMD feature extraction technique, a signal is assumed to contain multiple simple intrinsic modes of oscillations. The key feature of this method is that it can decompose complex signals into IMF components. Each decomposed IMF component contains the local characteristic signals of different time scales of the original signal.

The specific process of the EMD is as follows.

Step 1: For a signal $X(t)$, cubic spline interpolation is applied to connect all the maximum and minimum points of the signal to form upper and lower envelope lines. The mean value $m_1$ of the upper and lower envelope lines is obtained. Finally, the following operation is performed:

$$h_1 = X(t) - m_1 \tag{1}$$

Step 2: $h_1$ is taken as a new signal $X(t)$, and the processing in Step 1 is carried out. This is repeated until $h_i$ meets the conditions (1) and (2) in the definition of the IMF. At this time, $h_i$ can be considered the first-stage IMF, typically recorded as $c_1$.

Step 3: $c_1$ is separated from the original signal $X(t)$ to obtain the following equation:

$$r_1 = X(t) - c_1 \tag{2}$$

Using $r_1$ as a new signal, we repeat Steps 1 and 2 until the obtained $r_n$ can no longer generate a new IMF.

After the above decomposition process, the final decomposition mathematical expression of the signal $X(t)$ is

$$X(t) = \sum_{i=1}^{n} c_i(t) - r_n(t) \tag{3}$$

In the above formula, $c_i$ is the *ith* IMF, and $r_n$ is the residual term. Fig. 1 shows the flowchart of the EMD process.

Notably, to apply the EMD to a signal, the signal must satisfy the following conditions: it should have at least one high and one low extreme point; the characteristic time scale should be equal to the interval between adjacent pole value points; if the signal has no pole and only an inflection point, it is necessary to differentiate the signal before decomposition to obtain its extreme point, and then integrate to obtain the component.

In the EMD process, the first decomposed $IMF_1$ is the highest frequency component, and the frequencies of the $IMF_2$, $IMF_3$, $\cdots$, and $IMF_n$ components decrease in turn.

The EMD process is essentially a layer-by-layer decomposition process. Huang *et al.* found that if there are too many decomposition levels, the IMF component will become a pure frequency modulation signal without restrictions, making it no longer have any practical physical significance. Therefore, Huang *et al.* put forward a stopping rule for the decomposition process.

$$S = \sum_{t=0}^{T} \frac{\left|h_{1(k-1)}(t) - h_{1k}(t)\right|^2}{h_{1k}^2(t)} \tag{4}$$

In the above formula, $h_{1(k-1)}$ and $h_{1k}$ are two consecutive time series in the decomposition process. Huang *et al.* pointed out that when the $S$ value is set between 0.2 and 0.3, the result of the EMD decomposition is the best [24], [25].

In this paper, the EMD method is used to analyze the sample signals of each group of mobile robotic roller bearings during the experiment. The sample vibration signals are decomposed into multi-order IMF components.

The vibration information of the mobile robotic roller bearing is mainly decomposed into $IMF_1$–$IMF_n$ components. Because of the limited information contained in the higher-order components, the $IMF_1$–$IMF_8$ components are

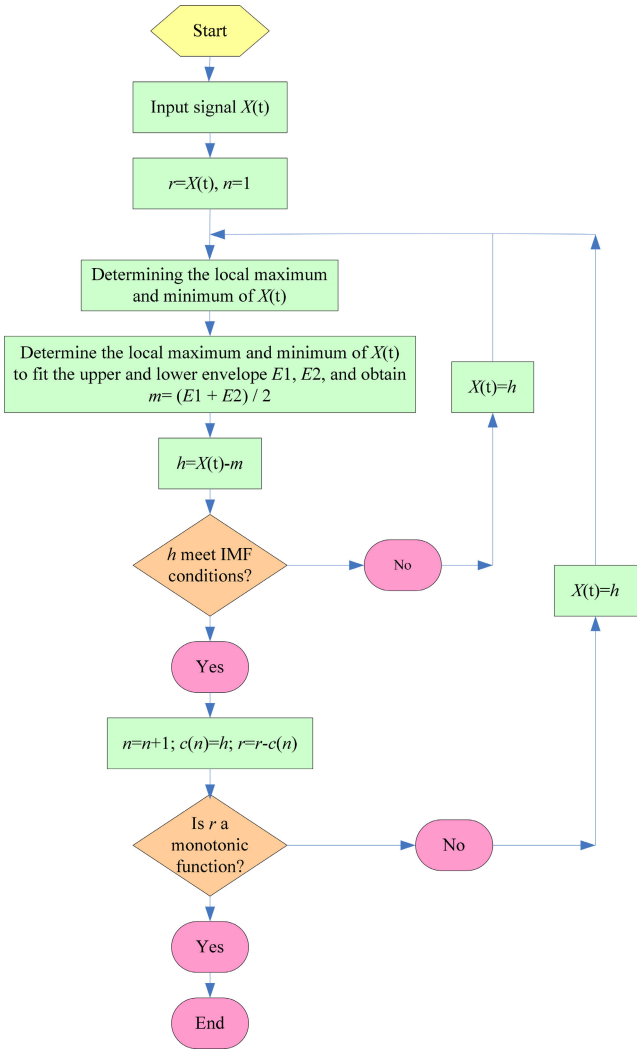


**FIGURE 1.** Flowchart of the EMD decomposition.

used as the features of the mobile robotic roller bearing signal in this study.

Fig. 2 shows the EMD composition of the vibration signal of a mobile robotic roller bearing with an inner-race fault. The decomposition performed by EMD is represented in the eight IMFs plotted in Fig. 2, where the last row corresponds to the final residue. The signal $X(t)$ is decomposed into eight IMFs with different time scales, whereby the characteristics of the signal $X(t)$ can be represented in different resolution ratios.

However, the signal sequence of each order is still long; this is difficult to be classified as the feature values. Therefore, the kurtosis values of each IMF component are used to represent the information contained in the components.

Kurtosis [26] is a dimensionless parameter used in statistics. It reflects the distribution characteristics of the vibration signal and describes the peak degree of vibration waveform. Its calculation formula is

$$K = \frac{\sum_{i=1}^{N} (x_i - u)^4}{\sigma^4 N} \tag{5}$$

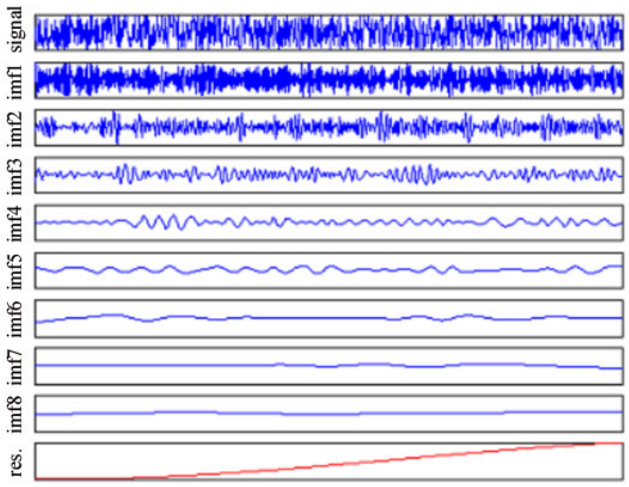


**FIGURE 2.** EMD of a vibration signal of a mobile robotic roller bearing with inner-race fault.

where $N$ is the length of the vibration sequence, $x_i$ is the amplitude of the ith point, $u$ is the average value of each $x$, and $\sigma$ is its standard deviation.

In the fault diagnosis of mobile robotic roller bearings, any change in the bearing speed or load has no effect on the kurtosis value of the signal; however, when the bearing surface is damaged, the kurtosis value will change significantly.

In other words, the kurtosis value of each IMF component of a mobile robotic roller bearing signal describes the vibration distribution features of the component signal and the strength of the fault impact signal in each order component.

Therefore, the kurtosis value vectors $[K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8]$ of the $IMF_1$–$IMF_8$ components are used as the feature values of the sample vibration signal.

## III. SUPPORT VECTOR MACHINE LEARNING FOR ROLLER BEARING FAULT DIAGNOSIS PROCESS

### A. SUPPORT VECTOR MACHIME

Corinna Cortes and Vapnik developed the currently used SVM technique to minimize the VC dimension [27]–[31]. The hyperplane, which separates data ($x \in R^n$) of two classes, can be expressed as follows.

$$x \cdot w - b = 0 \tag{6}$$

where $w$ is the weight vector $w \in R^n$, and $b$ is the offset $b \in R^n$.

The soft margin SVM algorithm was developed for cases wherein data cannot be linearly separated. The optimal problem solved by the soft margin SVM is given as follows.

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{7}$$

If $\alpha_i^*$ is the optimal value, we have:

$$\omega^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i \tag{8}$$

In this optimal problem, $D = \{(x_i, y_i)|i = 1, 2, \ldots, n, x \in R^n\}$ is the training dataset, and $y \in [-1, +1]$ is the target value of the training dataset.

For linear unclassified conditions, the hyper-plane can be transferred to solve the problem, as follows.

$$\Phi(\omega, \xi) = \frac{1}{2}(\omega \cdot \omega) + C\left(\sum_{i=1}^{n} \xi_i\right) \tag{9}$$

where the parameter $\xi_i$ is the slack variable, and parameter $C$ is called the punitive parameter.

Selecting an optimum value for $C$ is important to improve the performance of the SVM identifier. The SVM identifier is overfitted when $C$ is set too high, whereas it will have a large training error if $C$ is set too small. The parameter $C$ is typically estimated from the training data using a cross-validation procedure or other methods [32]–[34]. After pre-processing, $C$ is estimated from the training data [35]–[38].

### B. ROLLER BEARING INTELLIGENT FAULT DIAGNOSIS PROCESS

Fig. 3 shows the schematic of the experimental equipment used for the intelligent fault diagnosis of mobile robotic roller bearings. The shaft speed is approximately 600 r/min. The sample frequency is 10 Hz, and the data length is set to 1024 points.

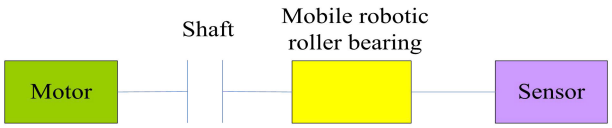


**FIGURE 3.** Schematic of experimental mobile robotic equipment.

Based on the frequency of the mobile robotic roller bearing signal, we can determine whether the bearing will fail.

The frequencies corresponding to the inner race, outer race, and roller signals can be expressed as follows, respectively:

The passing frequency of inner raceway is

$$f_{ip} = Z\frac{\sin\alpha}{\sin\alpha + \sin\beta}|f_0 - f_i| \tag{10}$$

The passing frequency of outer raceway is

$$f_{op} = Z\frac{\sin\beta}{\sin\alpha + \sin\beta}|f_0 - f_i| \tag{11}$$

The rotation frequency of the rolling element is

$$f_{bc} = \frac{2\cos\alpha\sin\alpha\sin\beta}{\sin^2\alpha - \sin^2\beta}|f_0 - f_i| \tag{12}$$

where $f_0$ is the inner circle frequency, $f_i$ is the outer circle frequency, $Z$ is the number of gears, $\alpha$ and $\beta$ are the contact angles.

Fig. 4 shows the parallel EMD-SVM model for the intelligent fault diagnosis of mobile robotic roller bearings. As mentioned above, the vibration data of the roller bearing for training can be decomposed into eight IMFs using the EMD method. The kurtosis vectors of the $IMF_1$–$IMF_8$ components $[K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8]$ are then used as the feature values of the sample signal, and the features are classified using the parallel SVM model.

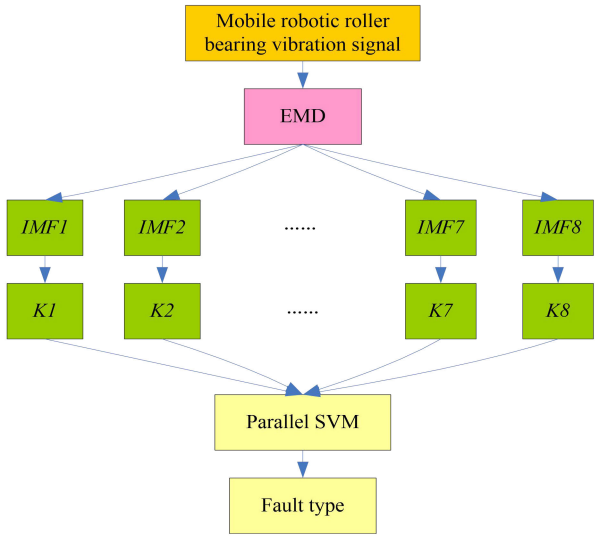


**FIGURE 4.** Parallel EMD-SVM intelligent fault recognition model.

The vibration signals of the mobile robotic roller bearings are mapped into eight IMFs, reduced to 8-dimentional kurtosis feature vectors, and finally classified by parallel SVM to achieve the so-called parallelism.

The parallel SVM is utilized to identify the various types of faults. The parallel SVM is trained using the radial basis function (RBF) kernel. The following are the three types of mobile robotic roller bearing faults: normal fault (Type 1), outer-race fault (Type 2), and inner-race fault (Type 3).

## IV. PARALLEL EMD-SVM BASED ON BIG DATA SPARK ON A MESOS IN A CLOUD COMPUTING FRAMEWORK

The parallel processing technology Spark [39]–[42] and EMD-SVM are combined to realize a parallel operation of the serial algorithm. Without compromising the classification accuracy, the classification efficiency of the algorithm can be significantly improved in a cloud computing framework.

### A. BRIEF INTRODUCTION OF THE COMBINED APPROACH

The algorithm process is divided into three stages: Map, combine, and reduce. The idea of divide and conquer is adopted: the algorithm uses the Spark framework to divide the dataset into several data sub-blocks and assign them to each thread executor. The local SVM is then trained in parallel, and the sub-classifiers are integrated. The sub-classifier trained by a data sub-block is local, making it necessary to retrain the integrated global classifier. Because the parallel training of the executor speeds up the convergence of the classifier, the global classifier only needs to be fine-tuned in the reduce stage, thus saving operation time. This algorithm not only improves the operation speed, but also improves the

classification accuracy. This can be attributed to the adoption of a multi-channel design, where each thread optimally utilizes the segmented dataset sub-block to train the local part of the class sub-model, overcoming the drawback of the traditional serial SVM where the local information of the sample is not fully utilized, and improving the performance of the classifier to a certain extent.

### B. ALGORITHM DESIGN AND IMPLEMENTATION IN A CLOUD COMPUTING NETWORK

The first process is dataset segmentation. The data samples are uploaded to the Hadoop distributed file system (HDFS). Based on the characteristic dimension and the specified number of blocks, the dataset is converted to a resilient distributed dataset (RDD) through the segmentation rules in the partition class provided by Spark, and then distributed to the executors.

The number of partition blocks can be determined by the number of cluster nodes and machine learning performance. In this study, two executors are set for each worker, and there are 17 machines in the cluster, so the number of partition blocks is 36. Thereafter, each sub-block calls randomsplit() to randomly divide the dataset into training and test samples in the proportion of 7:3.

#### 1) MAP STAGE

Each executor establishes a classifier based on the objective function, calls the train() method, and trains the classifier iteratively through the data sub-block. To adapt to large-scale training samples, this study uses the stochastic gradient descent (SGD) algorithm to optimize the parameters. Only some parts of the samples are involved in the calculation in each iteration. This reduces the memory cost and time required.

A setter object is created to rewrite train(): the number of iterations (numIterations) is set to 1200 with a step size of 2, and the sample participation ratio in each iteration (miniFatchFraction) is set to 2. To prevent overfitting, the ridge regression *L2* regulation is introduced as the correction function, and the regulation factor is 0.2.

#### 2) COMBINE STAGE

The combine stage is a transition stage between the map and reduce stages. Through the combine object, all the data sub-blocks and local classifiers trained in the map stage are combined and sent to the reduce stage.

The combine stage is not a simple linear splicing stage. It calls repartition() by instantiating the object of ShuffledRdd class to realize the splicing after shuffling the original data and sub-classifier.

#### 3) REDUCE STAGE

The results returned in the combine phase are received, and predictpoint() is called to test the combined classifier. The predicted value of each record in the test sample is calculated to compare with the original data and obtain the error score rate. The error score rate is obtain from the receiver operating characteristic (ROC) curve. If the score is less than the threshold value, the classifier is unqualified, and the optimization is continued. If the score is greater than the threshold value, the classifier reaches the standard, and the test stage is completed. The prediction value, membership degree, classification accuracy, classification time, and ROC coefficient of each record in the test set of the global optimal classifier are printed. Fig. 5 shows the data flow of the parallel SVM based on Spark. Before training the model, we need to upload the dataset to the HDFS file storage system. The task scheduling system of the Spark cluster will create new tasks in the executor for each data subset after segmentation, and use the resource scheduler to allocate the computing resources to the corresponding task allocator. The algorithm performs layered parallel SVM training on the Spark cluster, cooperates with the local SVM until the best training is completed, and outputs the global optimal model to the HDFS file system.

### C. MESOS CLUSTER RESOURCE MANAGEMENT SYSTEM

#### 1) MESOS ARCHITECTURE

Considering the operation and maintenance costs, data sharing, resource utilization, and other factors, companies generally want to deploy different distributed computing frameworks (Hadoop and Spark) on the same cluster to realize resource sharing of the cluster.

However, they will interfere with each other because of their joint operation. Therefore, an effective resource isolation mechanism and a unified scheduling mechanism should be provided to avoid the inefficiency due to resource contention between tasks.

Mesos [43], [44] is an open-source distributed resource management system under Apache. It is called the kernel of the distributed system.

Mesos allows multiple frameworks to share resources in a fine-grained manner. The multiple frameworks can access the stored data on the same machine to achieve efficient data local utilization.

Fig. 6 shows the main components of the Mesos with a centralized architecture. A master node process can manage several slave node processes running on each machine node.

The calculation tasks of the calculation framework are run on each slave node. As shown in Fig. 6, Hadoop and MPI share the resources of the entire cluster.

Hadoop's compute task executor can not only monopolize a machine node to run one or more MapReduce tasks, but can also share a machine with the message passing interface (MPI) compute task executor, a parallel computing framework based on exchanging messages. However, under the resource control strategy of Mesos, the executors of Hadoop and MPI achieve fine-grained resource isolation without interference. ZooKeeper can be used to solve the single-point-of-failure problem of the primary node of Mesos.

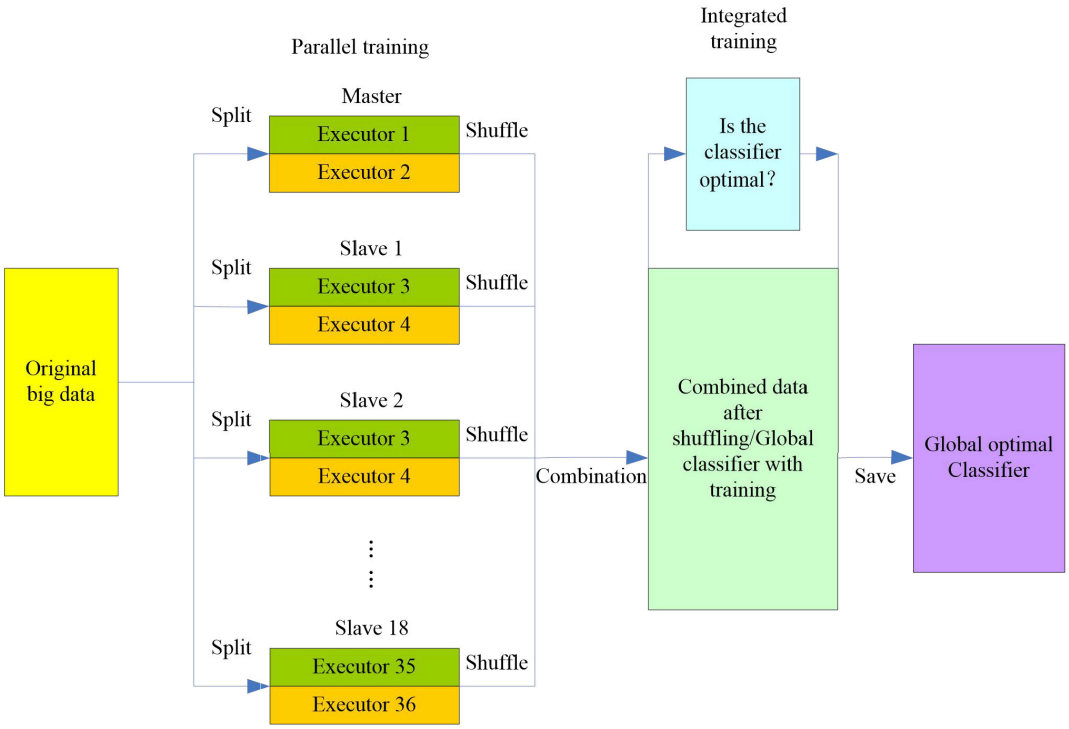


**FIGURE 5.** Parallel EMD-SVM dataflow based on Spark big data cloud computing software framework.

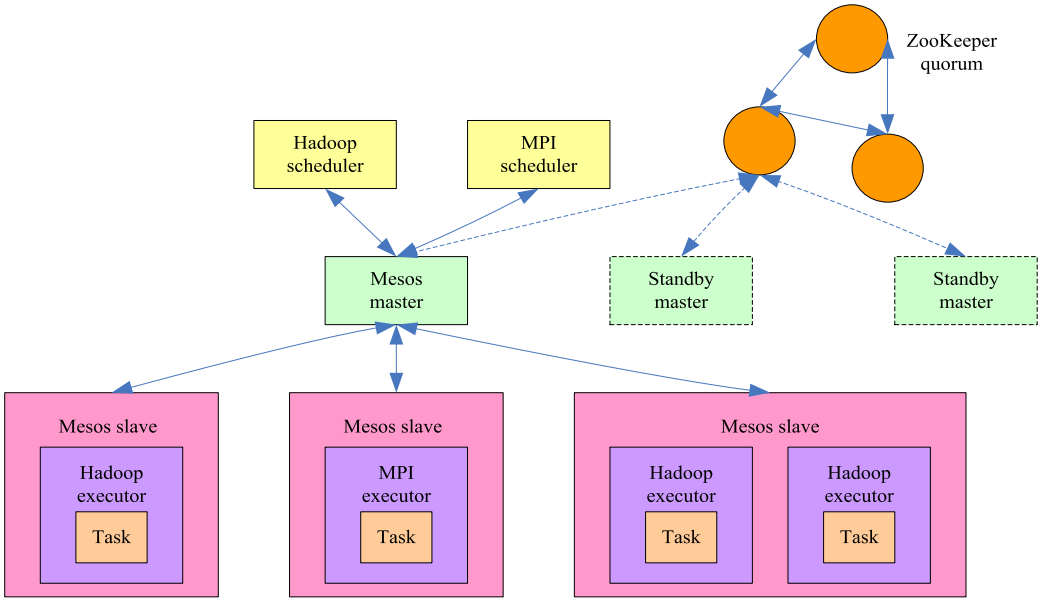


**FIGURE 6.** Software architecture of Mesos.

### 2) TASK RUNNING

The Spark big data processing framework is chosen as the test program. This section introduces the overall architecture of Spark running on Mesos as a big data cloud computing framework.

Fig. 7 shows the overall structure of Spark running on Mesos. Mesos (Cluster Manager in Fig. 7) replaces Spark's master process, i.e., Mesos is responsible for providing resources to Spark's calculation frame.

After receiving the resources pushed by Mesos, the Sparkcontext in the Spark big data cloud computing framework starts its own scheduling using the Round-Robin scheduling algorithm. Based on the resource requirements of each task (i.e., the resource requirements specified when

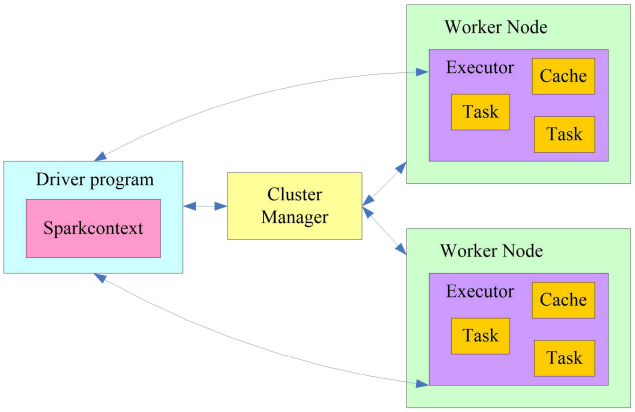


**FIGURE 7.** Overall architecture of Spark running on Mesos as a big data cloud computing software framework.

the user submits Spark), the resources provided by Mesos are allocated to the tasks, and the results of source and task matching are sent to Mesos (Cluster Manager).

Mesos further sends Spark's task information to the daemons on the corresponding nodes (Slave). After receiving the task information from the slave on the node, the executor of Spark is started with the specified resource constraint. The actuator of Spark starts one or more Spark tasks. At the same time, Spark's executor can communicate with Sparkcontext through the TCP protocol to report the status repair information of the task calculation process. From the perspective of Spark users, when users need to submit Spark program information, on the one hand, they need to specify the network address of the Mesos node daemons (Master), so that the Spark computing framework can be deployed to the Mesos cluster. On the other hand, they need to specify the memory capacity of each executor in the Spark configuration file.

A set of new resource measurement methods considering the fine-grained heterogeneity of hardware resources is defined in this paper. On the one hand, it is necessary to accurately measure the heterogeneous resources in the big data cloud computing platform. On the other hand, it is important to realize an optimal resource allocation on the big data cloud computing platform.

#### 3) FUNCTION OF MESOS

Mesos brings benefits to both developers and operation and maintenance (O&M) personnel. It can not only save infrastructure cost, but also bring convenience to the operation and maintenance team. It can help developers simplify the interface of the infrastructure, ultimately bringing huge benefits to the enterprise. Mesos can support multiple workloads and frameworks, improve resource utilization by flexibly sharing resources among frameworks, and help developers build efficient modern applications.

## V. EXPERIMENTAL RESULT AND DISCUSSION

### A. EXPERIMENTAL ENVIRONMENT

The experiment involved building 18 virtual machines on six physical machines as nodes of the cluster environment in a cloud computing framework, totaling 18 nodes including 1 master node and 17 slave nodes. The processors of the six physical machines have eight cores with a running frequency of 3.6 GHz, a DDR3L memory of 128 GB, and a hard disk size of 1500 GB. The 18 virtual machines have 4 GB of memory and 450 GB of hard disk space. VirtualBox software is used for installation in the virtual machine. Table 1 lists the details of the software environment of the 18 virtual machines.

**TABLE 1.** Spark cluster machine software environment information.

| Computer | Operating System | Hadoop | Spark | jdk | scala | Anaconda | Python |
|---|---|---|---|---|---|---|---|
| Data1 | Ubuntu 18.04 | 3.2.1 | 3.0.0 | 10.0.2 | 2.0.6 | 3-5.2.0 | 3.6.8 |
| Data2 | Ubuntu 18.04 | 3.2.1 | 3.0.0 | 10.0.2 | 2.0.6 | 3-5.2.0 | 3.6.8 |
| Data3 | Ubuntu 18.04 | 3.2.1 | 3.0.0 | 10.0.2 | 2.0.6 | 3-5.2.0 | 3.6.8 |
| | | | ..... | | | | |
| Data18 | Ubuntu 18.04 | 3.2.1 | 3.0.0 | 10.0.2 | 2.0.6 | 3-5.2.0 | 3.6.8 |

### B. COMPARATIVE ANALYSIS WITH PREVIOUS WORK

Although the SVM is suitable for solving small samples, its performance in solving massive datasets is poor. To improve the processing performance for intelligent fault recognition, we applied a Spark programming mode to the nonlinear SVM algorithm, implemented a parallel SVM algorithm based on Spark, and compared it with two previous algorithms: MapReduce SVM [45]–[47] and SVM [48], [49].

The SVM model has two important parameters: $C$ and $\gamma$. Here, $C$ is the punitive parameter, and $\gamma$ is a parameter that comes with the RBF function when it is selected as the kernel. The dataset used in the experiment contains 568264 samples, including 287636 training samples and 280628 testing samples. Based on the data size of the dataset, the best $\gamma$ value obtained is 0.03 through parameter optimization.

Table 2 lists the average classification accuracies of fault recognition for mobile robotic roller bearings corresponding to different values of the parameter $C$. The number of iterations is 6840. Fig. 8 shows the curves of the average classification accuracy.

The following conclusions can be drawn from the experimental results. As shown in Table 2 and Fig. 8, when $C$ is 4000, the average classification accuracies of the SVM, MapReduce SVM, and Spark SVM are 91.63, 93.82, and 94.96%, respectively. Compared with the SVM and MapReduce SVM, the Spark SVM has a higher average classification accuracy, thus demonstrating that the current Spark SVM outperforms the previous algorithms.

### C. PERFORMANCE OF PARALLEL EMD-SVM IN A BIG DATA CLOUD COMPUTING NETWORK

To compare the intelligent fault diagnosis results of different techniques, two other classifiers, namely DBN and RBFNN, were used in addition to the multi-class SVM

**TABLE 2.** Average classification accuracies of fault recognition corresponding to different values of the parameter *C*.

| *C* | SVM | MapReduce SVM | Spark SVM |
|---|---|---|---|
| 4 | 88.56 | 89.31 | 89.79 |
| 40 | 89.74 | 90.63 | 91.76 |
| 400 | 90.25 | 92.73 | 93.84 |
| 2000 | 90.78 | 92.86 | 93.24 |
| 4000 | 91.63 | 93.82 | 94.96 |
| 6000 | 90.38 | 92.51 | 93.89 |

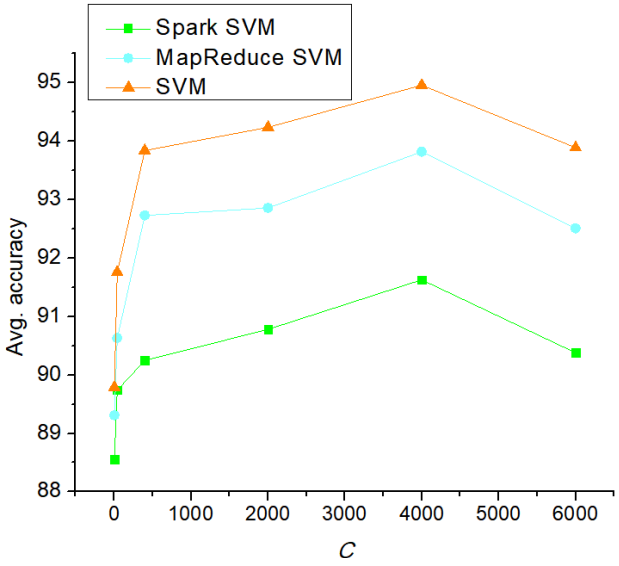


**FIGURE 8.** Average classification accuracy curve.

**TABLE 3.** Number of samples in each parallel machine learning sample set.

| Training set | Test set No. 1 | Test set No. 2 | Test set No. 3 | Test set No. 4 |
|---|---|---|---|---|
| 200000 | 300000 | 500000 | 800000 | 1000000 |

approach. Table 3 lists the details of the parallel machine learning sample set.

We selected ten training sets having 25000, 50000, 75000, 100000, 125000, 150000, 175000, 200000, 225000, and 250000 samples, respectively. Another 500000 samples were selected as the testing set. The sensitivity of the parallel EMD-SVM fault diagnosis to the nature of the sample is evident in Fig. 9, where the parallel EMD-SVM fault diagnosis results based on different training sets are varied in terms of the accuracy. The fault diagnosis results show that the average accuracy is related to the size of the training set (Fig. 9). For the fault diagnosis made by the parallel EMD-SVM, the average accuracy is highest when the training set size is 200000 and lowest when the training set size is 25000; the difference in the average accuracies is 8.4%. Under each training set size, the parallel EMD-SVM is overall the most accurate classifier, with the average accuracy being statistically different from those of the parallel EMD-DBN and parallel EMD-RBFNN classifiers. The parallel EMD-DBN gives better identification results than the parallel RBFNN. The classification made by the parallel EMD-RBFNN algorithm appears to be least sensitive to the training set size, with an average accuracy increasing from 82.4 to 87.3%.

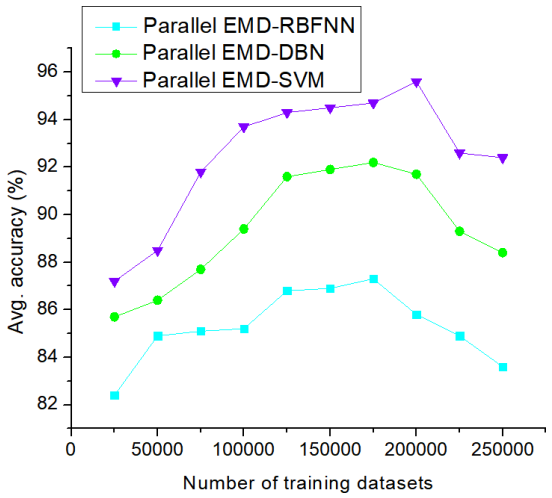


**FIGURE 9.** Average accuracy (%) of parallel machine learning classification method under different training set sizes.

Fig. 10 shows the average recognition times of the parallel machine learning classification methods under each training set size. The average recognition time of the parallel

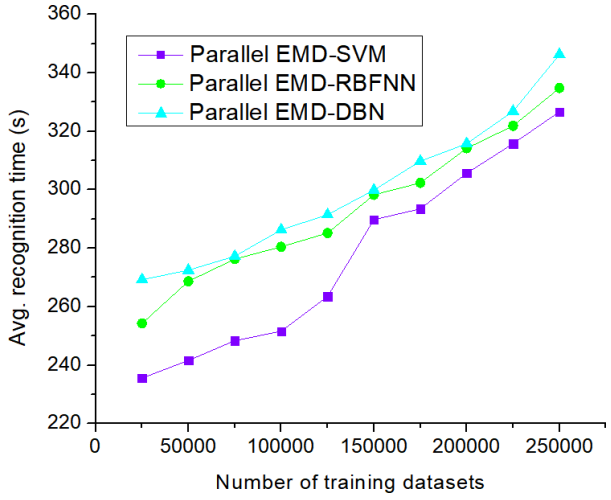


**FIGURE 10.** Average recognition time (s) of the parallel machine learning classification methods under different training set sizes.

EMD-SVM is lower than those of the parallel EMD-RBFNN and parallel EMD-DBN. As the dataset number increases from 25000 to 250000, the average recognition time of the parallel EMD-SVM increases from 235.6 to 326.7 s. The parallel EMD-RBFNN slightly outperforms the parallel EMD-DBN in terms of the time required.

To evaluate the influence of different test set sizes on the accuracy of SVM, we fixed the size of the training set to 200000. This value was chosen because the training set size of 200000 gives a good fault diagnosis accuracy (approximately 95.6%), as shown in Fig. 9.

We obtained five sample sets, one for training and the rest for testing, as listed in Table 3.

In the training stage, the parameters of the identifier are appropriately selected by trial and error. The Gaussian RBF kernel was used for training and testing the SVM, and the values of the variance of the Gaussian ($\sigma^2$) and punitive parameter ($C$) were set to 0.04 and $2.54 \times 10^4$, respectively. The final key incluencing factors are the variance of the Gaussian $\sigma^2$ and punitive parameter $C$. The training time required for 200000 patterns is 426.58 s. Table 4 lists the identification results under the test sets. Clearly, the intelligent fault diagnosis results are accurate.

**TABLE 4.** Parallel machine learning test results for four test sets in a cloud computing network.

| | Test set No. 1 | Test set No. 2 | Test set No. 3 | Test set No. 4 |
|---|---|---|---|---|
| Accuracy ($\frac{correct}{total} \times 100\%$) | 94.33 | 94.80 | 95.75 | 95.23 |
| Time (s) | 453.46 | 527.62 | 692.86 | 741.53 |

### D. ANTI-NOISE PERFORMANCE OF PARALLEL EMD-SVM IN A NOISY CONDITION

To further evaluate the validity of the identifier when the input is corrupted by noise, we added a 15-60 dB normally distributed noise to the test data and performed the test again. The results are good, as listed in Table 5.

**TABLE 5.** Parallel machine learning test results for four test sets in a noisy condition in a big data cloud computing framework (%).

| | dB | Test set No.1 | Test set No.2 | Test set No.3 | Test set No.4 |
|---|---|---|---|---|---|
| Accuracy ($\frac{correct}{total} \times 100\%$) | 15 | 94.23 | 94.60 | 95.38 | 94.80 |
| | 30 | 93.33 | 93.40 | 95.13 | 94.30 |
| | 45 | 92.67 | 93.40 | 94.63 | 93.90 |
| | 60 | 92.33 | 92.60 | 93.88 | 93.20 |

To test the performance of the proposed technique in different noisy conditions, test sets containing noise with SNR values ranging from 15 to 60 dB were considered. The equation used to calculate the SNR is as follows [50]:

$$SNR = 10 \log \left( \frac{P_s}{P_n} \right) dB \tag{13}$$

where $P_s$ is the power (variance) of the signal, and $P_n$ is that of the noise.

Table 5 lists the overall fault diagnosis performance under different test sets in a noisy environment. As listed in Table 5, the proposed parallel EMD-SVM [51], [52] machine learning [53]–[56] method has a robust anti-noise performance, with high fault diagnosis accuracy in noisy conditions.

### E. CLASSIFICATION PERFORMANCE COMPARISON OF PARALLEL EMD-SVM AND SERIAL EMD-SVM

A total of 68942 fault patterns were used in the experiment. Based on the number of randomly selected samples, they were divided into four groups: sample_1, sample_2, sample_3, and sample_4. Seventy percent of each group of samples is used as the training set and the rest as the test set. Table 6 lists the details of the samples.

**TABLE 6.** Sample groups.

| Sample name | Number of data samples | Number of training samples | Number of testing samples |
|---|---|---|---|
| Sample_1 | 6854 | 4798 | 2056 |
| Sample_2 | 9572 | 6700 | 2872 |
| Sample_3 | 25894 | 18126 | 7768 |
| Sample_4 | 46486 | 32540 | 13946 |
| Sample_5 | 68942 | 48259 | 20683 |

The improved parallel EMD-SVM and serial EMD-SVM were tested under the same conditions. Through processing four groups of fault pattern datasets with increasing sample number, the classification accuracy and classification time of the two are obtained, and the influence of sample size on the two algorithms is analyzed.

As listed in Table 7, for processing the four groups of data samples of different sizes, the classification accuracy of the parallel EMD-SVM based on Spark is higher than that of the serial EMD-SVM. Because the parallel EMD-SVM adopts a multi-channel design and dataset segmentation, each thread can use a part of the segmented dataset carefully, overcoming the drawback observed in the traditional serial EMD-SVM.

**TABLE 7.** Experimental classification accuracy of two algorithms In a big data cloud computing framework (%).

| | Parallel EMD-SVM | | Serial EMD-SVM | |
|---|---|---|---|---|
| Sample | Outer-race fault | Inner-race fault | Outer-race fault | Inner-race fault |
| Sample_1 | 94.6 | 92.7 | 92.6 | 90.4 |
| Sample_2 | 85.9 | 81.5 | 83.2 | 80.2 |
| Sample_3 | 82.8 | 78.3 | 80.4 | 73.9 |
| Sample_4 | 81.7 | 79.4 | 82.5 | 77.3 |
| Sample_5 | 82.4 | 79.6 | 81.3 | 75.7 |

Compared with the serial EMD-SVM, which can train the classifier directly from all the samples, the learning effect is better.

As listed in Table 8, when the size of the test dataset is small, the processing speed of the parallel EMD-SVM is largely the same, and the efficiency of the serial EMD-SVM is slightly higher than that of the parallel EMD-SVM. This is because there are too few samples, and Spark clusters spend too much time on dataset segmentation and task scheduling, so the effect of parallel EMD-SVM is not significantly higher than that of serial EMD-SVM. When the number of test samples increases to a certain extent, the classification time of the serial EMD-SVM increases significantly, while that of the parallel EMD-SVM increases steadily. When the scale of the test samples is large, the parallel processing effect of the Spark clusters is significant, saving the running time of the algorithm overall.

**TABLE 8. Test classification time of two algorithms In a big data cloud computing framework (s).**

| Sample | Parallel EMD-SVM | Serial EMD-SVM |
|---|---|---|
| Sample_1 | 84.9 | 83.4 |
| Sample_2 | 95.1 | 94.7 |
| Sample_3 | 149.3 | 158.7 |
| Sample_4 | 186.2 | 226.8 |
| Sample_5 | 258.7 | 336.2 |

### F. AVERAGE RECOGNITION TIME PERFORMANCE EVALUATION OF DIFFERENT BIG DATA CLOUD COMPUTING FRAMEWORKS

Hadoop, Spark, and Storm are the three most important distributed computing systems. Hadoop adopts a MapReduce distributed computing framework and is equipped with an HDFS distributed file system based on the Google File System (GFS) and a Hadoop Database (HBase) data storage system based on BigTable. Spark is another important distributed computing system, with some architectural improvements on the basis of Hadoop. Spark's distributed computing based on the MapReduce algorithm has the advantages of Hadoop's MapReduce. Spark is an open-source cluster computing system based on memory computing, intended for analyzing data more quickly. Spark can be better applied to MapReduce algorithms that require iteration, such as in the case of data mining and machine learning. The main difference between Spark and Hadoop is that Hadoop uses a hard disk to store data, whereas Spark uses memory to store data; therefore, Spark can provide faster computation. However, Spark cannot be used to process data that need to be saved for a long time because data will be lost when the memory is powered off. The storm distributed computing system provides real-time computing features based on Hadoop; it can process large data streams in real time. Unlike Hadoop and Spark, Storm does not collect and store data—it receives and processes data in real time through the network directly, and then sends back the results in real time directly through the network.

The performance of the proposed algorithms is evaluated in a Spark cluster comprising one master and 17 slave nodes. The simulation results are compared with those of the existing methods. In experimental observation, the average recognition time is an important parameter for evaluating the efficiency of algorithms, as shown in Fig. 11. In this model, the average recognition time is defined as the sum of the scheduling, execution, and transfer times required for recognition on different nodes. The results show that when the number of nodes is high, the average recognition time of the proposed Spark model is lower than those of Storm and Hadoop. For a small number of nodes, the processing time of the proposed Spark model is less than that of Hadoop, but close to that of Storm. However, the increase in the number of slave nodes does not increase the recognition time due to data sparsity. The saturation point of the average recognition time is detected after 13 slave nodes. Therefore, the average recognition time for mobile robotic intelligent faults is directly proportional to the scheduling, execution, and data transmission times.

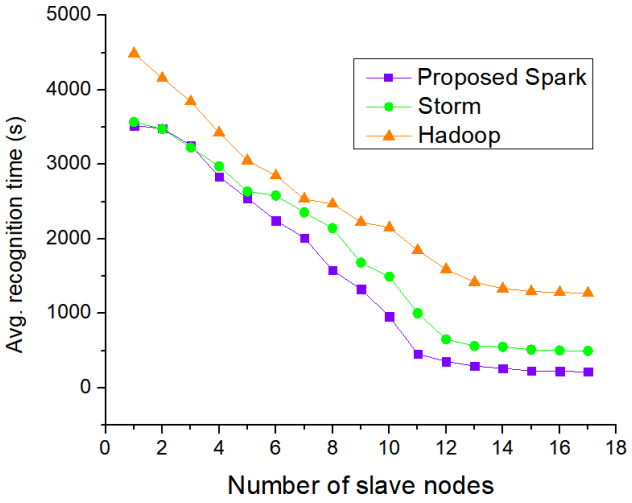


**FIGURE 11. Average recognition time for different cluster sizes in different big data cloud computing frameworks.**

Fig. 12 shows the average recognition time for robotic faults in different computing frameworks under different input data sizes. For small data sizes, i.e., less than 0.30 GB ($\cong$ 300 MB), the average recognition time of the proposed Spark framework is close to that of Storm. However, a noticeable average recognition time gap is observed for data sizes $>$ 0.30 GB, and this trend continues until it reaches up to 1.2 GB. Hence, the proposed Spark framework has a lower average recognition time than Storm and Hadoop.

### G. EVALUATION OF AVERAGE RECOGNITION TIME AND AVERAGE ACCURACY IN DIFFERENT DISTRIBUTED DEPLOYMENT MODES OF SPARK

Currently, Apache Spark supports three distributed deployment modes: standalone, Spark on Mesos [44], [57], and

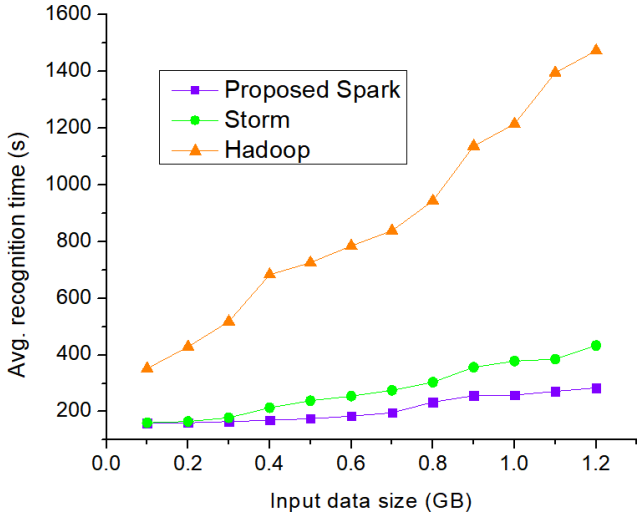


**FIGURE 12.** Average recognition time (s) of different big data cloud computing frameworks under different input data sizes.

Spark on YARN [58]. The first one is similar to the one adopted by MapReduce 1.0. It internally implements fault tolerance and resource management. The latter two are future development trends. Some parts of fault tolerance and resource management are completed by a unified resource management system. Spark runs on a common resource management system, so it can share a cluster resource with other computing frameworks such as MapReduce. The main advantages of using Spark are the reduced operation and maintenance costs and improved resource utilization.

It would be more flexible and natural to run Spark on Mesos than on YARN. In the Spark on Mesos environment, users can choose one of the following two scheduling modes to run their own applications: coarse-grained mode or fine-grained mode. In the coarse-grained mode, the running environment of each application consists of a driver and several executors. Each executor takes up several resources and can run multiple tasks internally (corresponding to several "slots"). Before each task of the application is formally run, all the resources in the running environment need to be applied for, and these resources should always be occupied during the running process. Even if these resources are not used, they will be recycled after the final program run. For the fine-grained mode, when the application starts, the executor will be started first, but each executor occupies only the resources needed for its own operation and does not need to consider the tasks to be run in the future. Thereafter, Mesos will dynamically allocate resources for each executor. Each time some resources are allocated, a new task can be run. After a single task runs, the corresponding resources can be released immediately. Each task will report its status to the Mesos slave and Mesos master for better fine-grained management and fault tolerance.

Table 9 lists the average recognition accuracy in the different distributed deployment modes of Spark under different

**TABLE 9.** Average recognition accuracy (%) in different distributed deployment modes of Spark under different cluster sizes.

| Number of nodes | Spark on Mesos (Fine-grained mode) | | | Spark on Mesos (Coarse-grained mode) | | | Spark on YARN | | | Stand-alone method | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg.*P* | Avg.*R* | Avg.*F1* | Avg.*P* | Avg.*R* | Avg.*F1* | Avg.*P* | Avg.*R* | Avg.*F1* | Avg.*P* | Avg.*R* | Avg.*F1* |
| 1 | 88.6 | 93.5 | 91.6 | 87.4 | 92.6 | 90.5 | 86.4 | 91.3 | 90.3 | 86.3 | 91.1 | 89.8 |
| 2 | 89.1 | 92.7 | 91.8 | 88.3 | 92.3 | 90.8 | 87.4 | 90.3 | 90.6 | 86.9 | 90.9 | 90.2 |
| 3 | 88.7 | 91.5 | 92.1 | 88.6 | 91.5 | 91.6 | 87.9 | 90.8 | 91.5 | 87.5 | 89.8 | 90.1 |
| 4 | 89.6 | 93.8 | 92.4 | 89.3 | 90.4 | 92.3 | 88.7 | 91.3 | 91.8 | 88.3 | 87.4 | 89.2 |
| 5 | 90.3 | 91.6 | 91.7 | 90.1 | 91.4 | 92.9 | 88.8 | 92.1 | 91.9 | 88.1 | 88.7 | 90.4 |
| 6 | 91.5 | 92.3 | 92.1 | 90.7 | 91.7 | 93.1 | 89.9 | 91.5 | 90.4 | 88.7 | 89.3 | 91.4 |
| 7 | 91.8 | 92.7 | 92.8 | 90.8 | 92.1 | 91.7 | 90.3 | 92.3 | 91.8 | 88.4 | 90.2 | 91.9 |
| 8 | 91.9 | 93.8 | 92.2 | 91.6 | 93.6 | 92.6 | 91.3 | 92.5 | 92.4 | 89.3 | 91.4 | 91.2 |
| 9 | 92.4 | 93.9 | 92.8 | 92.7 | 92.4 | 92.5 | 92.3 | 91.7 | 92.2 | 90.2 | 91.5 | 91.4 |
| 10 | 93.2 | 93.7 | 92.6 | 93.1 | 92.9 | 92.3 | 92.4 | 92.1 | 92.2 | 90.3 | 91.7 | 92.1 |
| 11 | 94.8 | 93.8 | 93.1 | 93.9 | 93.1 | 92.6 | 91.8 | 92.6 | 91.5 | 90.8 | 92.1 | 92.4 |
| 12 | 94.7 | 94.1 | 95.1 | 92.5 | 93.7 | 92.7 | 93.1 | 92.6 | 92.3 | 90.5 | 91.5 | 92.6 |
| 13 | 95.2 | 95.1 | 95.8 | 94.8 | 94.3 | 95.4 | 93.3 | 92.8 | 92.4 | 91.7 | 92.6 | 92.7 |
| 14 | 95.7 | 95.8 | 96.3 | 95.3 | 95.6 | 96.4 | 93.6 | 92.5 | 92.1 | 92.1 | 93.2 | 92.7 |
| 15 | 95.4 | 95.2 | 96.8 | 95.7 | 95.2 | 96.1 | 94.1 | 94.4 | 92.3 | 91.8 | 92.7 | 91.8 |
| 16 | 96.5 | 96.2 | 97.1 | 96.3 | 95.8 | 96.7 | 94.3 | 94.2 | 94.7 | 91.6 | 93.1 | 92.1 |
| 17 | 97.3 | 97.8 | 97.9 | 97.1 | 96.4 | 96.5 | 95.4 | 94.7 | 95.2 | 92.6 | 93.7 | 92.8 |

**TABLE 10.** Average recognition accuracy (%) in the different distributed deployment modes of Spark under different input data sizes.

| Input data Size (GB) | Spark on Mesos (Fine-grained mode) | | | Spark on Mesos (Corse-grained mode) | | | Spark on YARN | | | Stand-alone method | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg.*P* | Avg.*R* | Avg.*F1* | Avg.*P* | Avg.*R* | Avg.*F1* | Avg.*P* | Avg.*R* | Avg.*F1* | Avg.*P* | Avg.*R* | Avg.*F1* |
| 0.1 | 84.6 | 84.1 | 83.9 | 82.1 | 83.6 | 82.3 | 80.2 | 83.1 | 81.3 | 78.4 | 79.3 | 78.1 |
| 0.2 | 85.6 | 84.3 | 84.9 | 83.2 | 85.1 | 84.1 | 81.8 | 84.6 | 84.6 | 80.2 | 81.4 | 82.1 |
| 0.3 | 87.5 | 86.8 | 87.3 | 84.8 | 87.1 | 87.2 | 82.5 | 86.3 | 85.6 | 82.8 | 83.4 | 83.2 |
| 0.4 | 88.4 | 89.3 | 88.6 | 85.8 | 89.3 | 87.6 | 84.7 | 87.3 | 86.3 | 82.1 | 85.3 | 84.3 |
| 0.5 | 90.3 | 90.1 | 91.3 | 87.6 | 90.2 | 88.9 | 86.3 | 86.8 | 86.1 | 84.5 | 86.2 | 87.3 |
| 0.6 | 91.2 | 92.1 | 92.3 | 89.3 | 91.2 | 91.2 | 88.3 | 87.3 | 88.1 | 86.3 | 85.9 | 86.3 |
| 0.7 | 92.1 | 94.8 | 94.9 | 91.4 | 93.4 | 93.1 | 89.4 | 88.2 | 89.6 | 87.3 | 88.4 | 88.9 |
| 0.8 | 93.9 | 95.1 | 94.7 | 92.1 | 94.7 | 94.2 | 89.3 | 90.6 | 90.8 | 89.4 | 90.2 | 90.4 |
| 0.9 | 94.7 | 96.8 | 95.2 | 93.6 | 96.1 | 95.1 | 90.3 | 92.1 | 91.8 | 90.4 | 91.4 | 91.2 |
| 1.0 | 95.6 | 95.1 | 95.6 | 94.1 | 95.8 | 94.7 | 92.4 | 93.1 | 93.2 | 91.4 | 92.4 | 92.3 |
| 1.1 | 96.3 | 95.2 | 95.6 | 95.1 | 96.3 | 95.3 | 93.4 | 93.2 | 93.7 | 92.1 | 93.6 | 93.7 |
| 1.2 | 97.1 | 95.4 | 96.2 | 96.2 | 95.2 | 95.4 | 94.7 | 93.6 | 94.8 | 92.3 | 93.6 | 94.1 |

cluster sizes. As listed, when using 17 slave nodes, the average precision, average recall, and average *F1* score of Spark on Mesos in the fine-grained mode reach 97.3, 97.8, and 97.9%, respectively. The average recognition accuracy of intelligent fault identification with Spark on Mesos in the fine-grained mode under different cluster sizes is better than that of Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method.

Fig. 13 shows the average recognition times in the different distributed deployment modes of Spark under different cluster sizes. As shown in Fig. 13, when using 17 slave nodes, the average recognition times of Spark on Mesos in the fine-grained mode, Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method are 183.6, 149.5, 171.9, and 153.8 s, respectively. Clearly, the average recognition time for Spark on Mesos in the fine-grained mode under different cluster sizes is more than those for Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method.

Table 10 lists the average recognition accuracy in the different distributed deployment modes of Spark under different input data sizes. As listed in Table 10, when the input data size is 1.2 GB, the average precision, average recall, and average *F1* score of Spark on Mesos in the fine-grained mode reach 97.1, 95.4, and 96.2%, respectively. The average recognition accuracy of fault identification with Spark on Mesos in the fine-grained mode under different input data sizes is better than those of Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method.

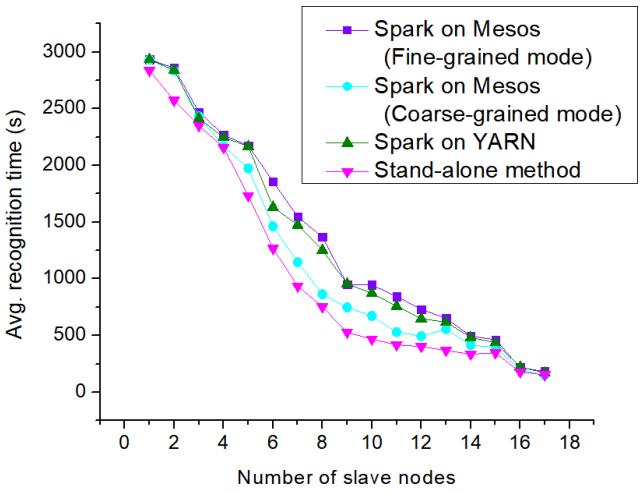


**FIGURE 13.** Average recognition time (s) in the different distributed deployment modes of Spark under different cluster sizes.

Mesos in the fine-grained mode can be chosen as the distributed deployment mode of Spark.

Fig. 14 shows the average recognition time in the different distributed deployment modes of Spark under different input

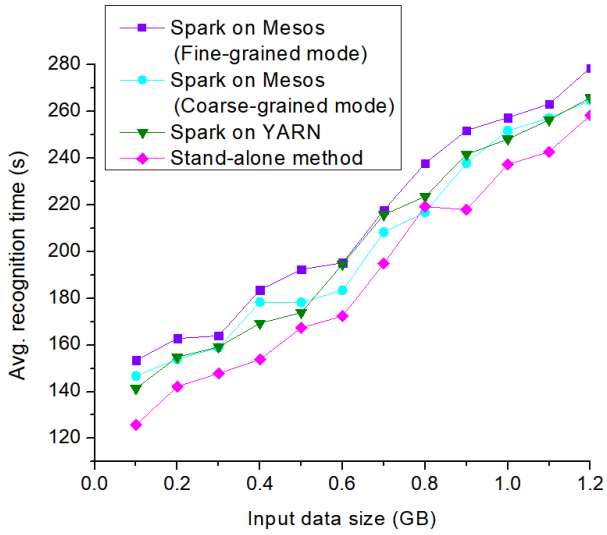


**FIGURE 14.** Average recognition time (s) in the different distributed deployment modes of Spark under different input data sizes.

data sizes. As shown in Fig. 14, when the input data size is 1.2 GB, the average recognition times of Spark on Mesos in the fine-grained mode, Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method are 278.7, 264.8, 265.9, and 258.4 s, respectively. Clearly, the average recognition time of intelligent fault identification with Spark on Mesos in the fine-grained mode under different input data sizes is more than those of Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method.

In summary, in terms of the time required and accuracy, Spark on Mesos in the fine-grained mode requires a slightly longer recognition time than Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method to obtain a better recognition accuracy. Therefore, Spark on Mesos in the fine-grained mode can be chosen as the distributed deployment mode of Spark.

## VI. CONCLUSIONS

This paper proposes a fault diagnosis technique for mobile robotic roller bearing faults using a parallel EMD-SVM machine learning method considering the non-stationary characteristics of the fault vibration signals. The original acceleration vibration signal was decomposed by EMD, and some IMF components were obtained. A parallel SVM served as the identifier, and the extracted energy features of the stationary IMFs were taken as input vectors to this network. Thus, faulty and non-faulty bearings could be distinguished. Under different training set sizes, the parallel EMD-SVM outperformed the parallel EMD-DBN and parallel EMD-RBFNN classifiers in terms of the classification accuracy, with the accuracy being statistically different from those of the other classifiers. To further evaluate the validity of the classifiers when the input is corrupted by noise, we added a 15-60 dB normally distributed noise to the test data and performed the test. The proposed parallel EMD-SVM machine learning technique exhibited a high fault diagnosis accuracy in this noisy condition as well. The method is expected to be useful for the fault diagnosis of mobile robotic roller bearings.

The SVM is a classification algorithm based on a large number of iterations, suitable for small sample environments. It exhibits a high complexity in processing large-scale datasets. This paper reports the design of a parallel EMD-SVM based on Spark in a cloud computing network. The algorithm is implemented on the Spark platform, which can segment large-scale datasets that can be used for the parallel training of classifiers. Mechanical fault classification tests were conducted on mobile robotic spherical bearings. The results showed that the parallel EMD-SVM based on Spark exhibits a higher classification accuracy than the traditional serial EMD-SVM, with a significantly lower average classification time, thus demonstrating the advantages of parallel processing of the big data Spark technology in cloud computing software frameworks for intelligent robots. The experimental results showed that the average accuracy of fault identification with Spark on Mesos in the fine-grained mode under different cluster and input data sizes is better than those of Spark on Mesos in the coarse-grained mode, Spark on YARN, and stand-alone method.

However, the proposed approach has certain limitations that should be addressed. For example, when there are too many support vectors in the dataset, the training efficiency of the algorithm is not high, and when there are too many data blocks, the accuracy decreases. There is no rigorous mathematical proof or theoretical derivation for the rationality of parallel design. In a subsequent study, we will perform a more in-depth research on a theoretical basis for the algorithm and the optimization of the training structure. Moreover, we will focus on the parameter optimization of the Spark platform, in order to obtain a better training effect and meet the requirements of scholars in training large-scale modular datasets with the parallel SVM.

## REFERENCES

[1] P. K. Sahoo, S. K. Mohapatra, and S.-L. Wu, "SLA based healthcare big data analysis and computing in cloud network," *J. Parallel Distrib. Comput.*, vol. 119, pp. 121–135, Sep. 2018.

[2] M. Wang, P. P. Jayaraman, E. Solaiman, L. Y. Chen, Z. Li, S. Jun, D. Georgakopoulos, and R. Ranjan, "A multi-layered performance analysis for cloud-based topic detection and tracking in big data applications," *Future Gener. Comput. Syst.*, vol. 87, pp. 580–590, Oct. 2018.

[3] D. Carnelossi Furlaneto, L. S. Oliveira, D. Menotti, and G. D. C. Cavalcanti, "Bias effect on predicting market trends with EMD," *Expert Syst. Appl.*, vol. 82, pp. 19–26, Oct. 2017.

[4] A. B. Das and M. I. H. Bhuiyan, "Discrimination and classification of focal and non-focal EEG signals using entropy-based features in the EMD-DWT domain," *Biomed. Signal Process. Control*, vol. 29, pp. 11–21, Aug. 2016.

[5] Y. Gong and L. Jia, "Research on SVM environment performance of parallel computing based on large data set of machine learning," *J. Supercomput.*, vol. 75, no. 9, pp. 5966–5983, Sep. 2019.

[6] G. Caruana, M. Li, and Y. Liu, "An ontology enhanced parallel SVM for scalable spam filter training," *Neurocomputing*, vol. 108, pp. 45–57, May 2013.

[7] A. Soualhi, K. Medjaher, and N. Zerhouni, "Bearing health monitoring based on Hilbert–Huang transform, support vector machine, and regression," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 1, pp. 52–62, Jan. 2015.
[8] J. H. Yan and L. Lu, "Improved Hilbert–Huang transform based weak signal detection methodology and its application on incipient fault diagnosis and ECG signal analysis," *Signal Process.*, vol. 98, pp. 74–87, May 2014.
[9] G. Cheng, Y. L. Cheng, and L. H. Shen, "Gear fault identification based on Hilbert–Huang transform and SOM neural network," *Measurement*, vol. 46, no. 3, pp. 1137–1146, Apr. 2013.
[10] Y. Yang, D.-J. Yu, and J.-S. Cheng, "A fault diagnosis approach for roller bearing based on IMF envelope spectrum and SVM," *Measurement*, vol. 40, nos. 9–10, pp. 943–950, Nov. 2007.
[11] Y. Lei, J. Lin, Z. He, and M. J. Zuo, "A review on empirical mode decomposition in fault diagnosis of rotating machinery," *Mech. Syst. Signal Process.*, vol. 35, nos. 1–2, pp. 108–126, Feb. 2013.
[12] F. Wu and L. Qu, "An improved method for restraining the end effect in empirical mode decomposition and its applications to the fault diagnosis of large rotating machinery," *J. Sound Vib.*, vol. 314, nos. 3–5, pp. 586–602, Jul. 2008.
[13] L. H. Smith, L. J. Hargrove, B. A. Lock, and T. A. Kuiken, "Determining the optimal window length for pattern recognition-based myoelectric control: Balancing the competing effects of classification error and controller delay," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 19, no. 2, pp. 186–192, Apr. 2011.
[14] P. Melin and O. Castillo, "A review on type-2 fuzzy logic applications in clustering, classification and pattern recognition," *Appl. Soft Comput.*, vol. 21, pp. 568–577, Aug. 2014.
[15] B. Boashash, N. A. Khan, and T. Ben-Jabeur, "Time–frequency features for pattern recognition using high-resolution TFDs: A tutorial review," *Digit. Signal Process.*, vol. 40, pp. 1–30, May 2015.
[16] C. Corinna and V. Vapnik, "Support-vector networks" *Mach. Learn.*, vol. 20, pp. 273–297, Sep. 1995.
[17] M. Mohammadi and A. Hezarkhani, "A comparative study of svm and rf methods for classification of alteration zones using remotely sensed data," *J. Mining Environ.*, vol. 11, no. 1, pp. 49–61, 2020.
[18] L. J. Cao and F. E. H. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1506–1518, Nov. 2003.
[19] N. Ramesh Babu and B. Jagan Mohan, "Fault classification in power systems using EMD and SVM," *Ain Shams Eng. J.*, vol. 8, no. 2, pp. 103–111, Jun. 2017.
[20] W. Y. Duan, Y. Han, L. M. Huang, B. B. Zhao, and M. H. Wang, "A hybrid EMD-SVR model for the short-term prediction of significant wave height," *Ocean Eng.*, vol. 124, pp. 54–73, Sep. 2016.
[21] S. Li, W. Zhou, Q. Yuan, S. Geng, and D. Cai, "Feature extraction and recognition of ictal EEG using EMD and SVM," *Comput. Biol. Med.*, vol. 43, no. 7, pp. 807–816, Aug. 2013.
[22] X. Liu, L. Bo, and H. Luo, "Bearing faults diagnostics based on hybrid LS-SVM and EMD method," *Measurement*, vol. 59, pp. 145–166, Jan. 2015.
[23] Y. Huang, D. Wu, Z. Zhang, H. Chen, and S. Chen, "EMD-based pulsed TIG welding process porosity defect detection and defect diagnosis using GA-SVM," *J. Mater. Process. Technol.*, vol. 239, pp. 92–102, Jan. 2017.
[24] N. E. Huang, Z. Shen, and S. R. Long, "A new view of nonlinear water waves: The Hilbert spectrum," *Annu. Rev. Fluid Mech.*, vol. 31, no. 1, pp. 417–457, Jan. 1999.
[25] R. Damaševičius, C. Napoli, T. Sidekerskienė, and M. Woźniak, "IMF mode demixing in EMD for jitter analysis," *J. Comput. Sci.*, vol. 22, pp. 240–252, Sep. 2017.
[26] S.-W. Fei, "Kurtosis forecasting of bearing vibration signal based on the hybrid model of empirical mode decomposition and RVM with artificial bee colony algorithm," *Expert Syst. Appl.*, vol. 42, no. 11, pp. 5011–5018, Jul. 2015.
[27] C. J. Zhu, K.-Y. Lam, J. K. Y. Ng, and J. Bi, "On the VC-dimension of unique round-trip shortest path systems," *Inf. Process. Lett.*, vol. 145, pp. 1–5, May 2019.
[28] C. Bazgan, F. Foucaud, and F. Sikora, "Parameterized and approximation complexity of partial VC dimension," *Theor. Comput. Sci.*, vol. 766, pp. 1–15, Apr. 2019.
[29] H. C. C. Carneiro, C. E. Pedreira, F. M. G. França, and P. M. V. Lima, "The exact VC dimension of the WiSARD *n*-Tuple classifier," *Neural Comput.*, vol. 31, no. 1, pp. 176–207, Jan. 2019.
[30] H. Hatami and Y. Qian, "Teaching dimension, VC dimension, and critical sets in latin squares," *J. Combinatorics*, vol. 9, no. 1, pp. 9–20, 2018.
[31] A. Munaro, "The VC-dimension of graphs with respect to *k*-connected subgraphs," *Discrete Appl. Math.*, vol. 211, pp. 163–174, Oct. 2013.
[32] C. Cao, R. L. Tutwiler, and S. Slobounov, "Automatic classification of athletes with residual functional deficits following concussion by means of EEG signal using support vector machine," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 16, no. 4, pp. 327–335, Aug. 2008.
[33] G. M. Foody and A. Mathur, "A relative evaluation of multiclass image classification by support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 6, pp. 1335–1343, Jun. 2004.
[34] Q. Wu, X. Shen, Y. Li, G. Xu, W. Yan, G. Dong, and Q. Yang, "Classifying the multiplicity of the EEG source models using sphere-shaped support vector machines," *IEEE Trans. Magn.*, vol. 41, no. 5, pp. 1912–1915, May 2005.
[35] X. Liang, L. Zhu, and D.-S. Huang, "Multi-task ranking SVM for image cosegmentation," *Neurocomputing*, vol. 247, pp. 126–136, Jul. 2017.
[36] V. K. Sharma and K. K. Mahapatra, "Visual object tracking based on sequential learning of SVM parameter," *Digit. Signal Process.*, vol. 79, pp. 102–115, Aug. 2018.
[37] J. Cervantes, F. Garcia-Lamont, L. Rodriguez, A. López, J. R. Castilla, and A. Trueba, "PSO-based method for SVM classification on skewed data sets," *Neurocomputing*, vol. 228, pp. 187–197, Mar. 2017.
[38] S. Sahran, D. Albashish, A. Abdullah, N. A. Shukor, and S. Hayati Md Pauzi, "Absolute cosine-based SVM-RFE feature selection method for prostate histopathological grading," *Artif. Intell. Med.*, vol. 87, pp. 78–90, May 2018.
[39] Z. Tang, A. Zeng, X. Zhang, L. Yang, and K. Li, "Dynamic memory-aware scheduling in spark computing environment," *J. Parallel Distrib. Comput.*, vol. 141, pp. 10–22, Jul. 2020.
[40] Y. Wang, Z. Gui, H. Wu, D. Peng, J. Wu, and Z. Cui, "Optimizing and accelerating space–time Ripley's K function based on Apache Spark for distributed spatiotemporal point pattern analysis," *Future Gener. Comput. Syst.*, vol. 105, pp. 96–118, Apr. 2020.
[41] H. Zhang, H. Huang, and L. Wang, "Meteor: Optimizing spark-on-yarn for short applications," *Future Gener. Comput. Syst.*, vol. 101, pp. 262–271, Dec. 2019.
[42] Z. Shmeis and M. Jaber, "A rewrite-based optimizer for spark," *Future Gener. Comput. Syst.*, vol. 98, pp. 586–599, Sep. 2019.
[43] S. López-Huguet, A. Pérez, A. Calatrava, C. de Alfonso, M. Caballer, G. Moltó, and I. Blanquer, "A self-managed mesos cluster for data analytics with QoS guarantees," *Future Gener. Comput. Syst.*, vol. 96, pp. 449–461, Jul. 2019.
[44] S. López-Huguet, I. Natanael, A. Brito, and I. Blanquer, "Vertical elasticity on marathon and chronos mesos frameworks," *J. Parallel Distrib. Comput.*, vol. 133, pp. 179–192, Nov. 2019.
[45] Z. H. You, J. Z. Yu, L. Zhu, S. Li, and Z. K. Wen, "A MapReduce based parallel SVM for large-scale predicting protein–protein interactions," *Neurocomputing.*, vol. 145, pp. 37–43, Dec. 2014.
[46] W. Zhao, T. Fan, Y. Nie, F. Wu, and H. Wen, "Research on attribute dimension partition based on SVM classifying and MapReduce," *Wireless Pers. Commun.*, vol. 102, no. 4, pp. 2759–2774, Oct. 2018.
[47] N. K. Alham, M. Li, Y. Liu, and M. Qi, "A MapReduce-based distributed SVM ensemble for scalable image classification and annotation," *Comput. Math. with Appl.*, vol. 66, no. 10, pp. 1920–1934, Dec. 2013.
[48] J. W. He, Y. Lyu, J. Han, and C. Wang, "An SVM approach for five-phase current source converters output current harmonics and common-mode voltage mitigation," *IEEE Trans. Ind. Electron.*, vol. 67, no. 7, pp. 5232–5245, Jul. 2020.
[49] F. Cheng, J. Chen, J. Qiu, and L. Zhang, "A subregion division based multi-objective evolutionary algorithm for SVM training set selection," *Neurocomputing*, vol. 394, pp. 70–83, Jun. 2020.
[50] H. B. He and J. A. Starzyk, "A self-organizing learning array system for power quality classification based on wavelet transform," *IEEE Trans. Power Delivery*, vol. 21, no. 1, pp. 286–295, Jan. 2006.
[51] E. Meng, S. Huang, Q. Huang, W. Fang, L. Wu, and L. Wang, "A robust method for non-stationary streamflow prediction based on improved EMD-SVM model," *J. Hydrol.*, vol. 568, pp. 462–478, Jan. 2019.
[52] K. Thirumala, S. Pal, T. Jain, and A. C. Umarikar, "A classification method for multiple power quality disturbances using EWT based adaptive filtering and multiclass SVM," *Neurocomputing*, vol. 334, pp. 265–274, Mar. 2019.
[53] L. Jollans, R. Boyle, E. Artiges, T. Banaschewski, S. Desrivières, A. Grigis, J.-L. Martinot, T. Paus, M. N. Smolka, H. Walter, G. Schumann, H. Garavan, and R. Whelan, "Quantifying performance of machine learning methods for neuroimaging data," *NeuroImage*, vol. 199, pp. 351–365, Oct. 2019.

[54] I. U. Din, M. Guizani, J. J. P. C. Rodrigues, S. Hassan, and V. V. Korotaev, "Machine learning in the Internet of things: Designed techniques for smart cities," *Future Gener. Comput. Syst.*, vol. 100, pp. 826–843, Nov. 2019.
[55] É. D. M. Silveira, I. C. Passos, J. Scott, G. Bristot, E. Scotton, L. S. T. Mendes, A. C. U. Knackfuss, L. Gerchmann, A. Fijtman, A. R. Trasel, G. A. Salum, and M. Kauer-Sant'Anna, "Decoding rumination: A machine learning approach to a transdiagnostic sample of outpatients with anxiety, mood and psychotic disorders," *J. Psychiatric Res.*, vol. 121, pp. 207–213, Feb. 2020.
[56] M. Sharif, M. Attique, M. Z. Tahir, M. Yasmim, T. Saba, and U. J. Tanik, "A machine learning method with threshold based parallel feature fusion and feature selection for automated gait recognition," *J. Organizational End User Comput.*, vol. 32, no. 2, pp. 67–92, Apr. 2020.
[57] J. Meyerson, "Ben hindman on apache mesos," *IEEE Softw.*, vol. 33, no. 1, pp. 117–120, Jan. 2016.
[58] D. Cheng, X. Zhou, P. Lama, J. Wu, and C. Jiang, "Cross-platform resource scheduling for spark and MapReduce on YARN," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1341–1353, Aug. 2017.

**GUANGMING XIAN** (Member IEEE) received the B.Eng., M.Eng., and Ph.D. degrees from the South China University of Technology, Guangzhou, China, in 1998, 2003, and 2007, respectively. From 2009 to 2011, his postdoctoral scientific project research was jointly carried out in the computer science and technology postdoctoral scientific research flow station of the South China University of Technology and the postdoctoral scientific research workstation of Guangzhou Tianhe Software Park Management Committee. The content of his Postdoctoral Research Report was the application of machine learning in financial time series prediction. He is currently an Associated Professor with the School of Software, South China Normal University, Foshan, China. He has published a series of papers in academic journals and international conferences. His current research interests include artificial intelligence, parallel machine learning, deep learning, big data, cloud computing, and mobile robot.