Hindawi
Complexity
Volume 2018, Article ID 8079697, 16 pages
https://doi.org/10.1155/2018/8079697

Research Article

Dynamic Prediction Research of Silicon Content in Hot Metal Driven by Big Data in Blast Furnace Smelting Process under Hadoop Cloud Platform

Yang Han,[1,2] **Jie Li**,[2,3] **Xiao-Lei Yang**,[1,2] **Wei-Xing Liu**,[2,3] **and Yu-Zhu Zhang**[2,3]

[1]*College of Science, North China University of Science and Technology, Tangshan 063210, China*
[2]*Tangshan Key Laboratory of Engineering Computing, North China University of Science and Technology, Tangshan 063210, China*
[3]*The Ministry of Education Key Laboratory with Modern Metallurgical Technology, North China University of Science and Technology, Tangshan 063210, China*

Correspondence should be addressed to Jie Li; lijie-2573017@163.com

Received 2 May 2018; Accepted 25 July 2018; Published 15 October 2018

Academic Editor: Kaoru Ota

In order to explore a dynamic prediction model with good generalization performance of the content of [Si] in molten iron, an improved SVM algorithm is proposed to enhance its practicability in the big data sample set of the smelting process. Firstly, we propose a parallelization scheme to design an SVM solution algorithm based on the MapReduce model under a Hadoop platform to improve the solution speed of the SVM on big data sample sets. Secondly, based on the characteristics of stochastic subgradient projection, the execution time of the SVM solver algorithm does not depend on the size of the sample set, and a structured SVM algorithm based on the neighbor propagation algorithm is proposed, and on this basis, a parallel algorithm for solving the covariance matrix of the training set and a parallel algorithm of the tth iteration of the random subgradient projection are designed. Finally, the historical production big data of No. 1 blast furnace in Tangshan Iron Works II was analyzed during 2015.12.01~2016.11.30 using the reaction mechanism, control mechanism, and gray correlation model in the process of blast furnace iron-making, an essential sample set with input $[x_1(k), x_2(k-3), x_3(k-3), \ldots, x_{18}(k), x_{19}(k-1)]$ and output $[\mathrm{Si}](k+1)$ is constructed, and the dynamic prediction model of the content of [Si] in molten iron and the dynamic prediction model of [Si] fluctuation in the molten iron are obtained on the Hadoop platform by means of the structure and parallelized SVM solving algorithm. The results of the research show that the structural and parallel SVM algorithms in the hot metal [Si] content value dynamic prediction hit rate and lifting dynamic prediction hit rate were 91.2% and 92.2%, respectively. Two kinds of dynamic prediction algorithms based on structure and parallelization are 54 times and 5 times faster than traditional serial solving algorithms.

1. Introduction

Blast furnace anterograde and hot metal quality are the primary goals of iron-making process control. The silicon content of hot metal is an important characterization parameter for slag quality, tapping temperature, and hot metal quality. The fluctuation degree of silicon content is also an important parameter for the operation of a blast furnace. Therefore, it is necessary to construct a precise dynamic prediction model of molten iron content in molten iron. The support vector machine (SVM) algorithm is a widely used machine learning algorithm. The core idea of this algorithm is the maximum interval theory [1] and the structural risk minimization principle [2]. The algorithm solves a convex quadratic programming problem by training the sample data and obtains the optimal hyperplane. For the SVM solution algorithm, the traditional Newton iteration method [3] and the interior point method [4] can be used; these two algorithms have perfect research theories and are widely used. However, all the information of the Hessian matrix must be searched globally, which greatly reduces the solution speed. Based on

this, relevant scholars have conducted in-depth research on SVM solving algorithms and proposed many classical algorithms, such as block algorithm, decomposition algorithm, and sequence minimization algorithm [5–7]; to some extent, although these algorithms have improved the solving efficiency of SVM, the amount of data generated in the era of big data is extremely large and constantly escalating, making it impossible to apply classic serial SVM solving algorithms for big data sample sets. From the situation is the bottleneck of the SVM set by the calculation of memory and operation time.

Based on the bottleneck of the SVM solution algorithm, relevant scholars have developed the parallelization strategy of the algorithm according to the iterative principle and convergence speed of the algorithm. In many parallel computing environments, cloud computing as a mainstream parallel computing research platform is extended by distributed computing [8], parallel computing [9], grid computing [10], and virtualization [11]. Hadoop is an open-source project under the Apache Software Foundation, which provides a highly reliable and scalable environment for distributed computing. Its most convenient application is the ability to build big-scale cluster systems on PCs. Because the integrated learning method [12] obtains different base classifiers through a specific learning method and then determines the final classification results according to some integration strategy for the results of each base classifier, a better classification result than a single classifier can be obtained, and MapReduce as one of the cores of the Hadoop distributed system is mainly responsible for the decomposition and integration of data and computing modules. Therefore, it is natural to develop integrated learning algorithms on the Hadoop cloud computing platform.

SVM, as a machine learning algorithm with a solid foundation of mathematical theory and excellent generalization performance, should use the Hadoop cloud computing platform and parallel computing to break through its bottleneck in the processing of big data sets and promote the application scope of the algorithm.

Xiaole et al. [13] performed the partitioned parallel design for the classical SVM serial interior point solution algorithm; Feng et al. [14] applied the self-adaptive parallel subgradient projection algorithm to reconstruct the compressed sensing signal and accelerated the convergence of the algorithm by adjusting the mechanism of the expansion coefficient; Dong et al. [15] improved the efficiency of the SVM algorithm by using the matrix LDLT parallel decomposition method; and Zhao et al. [16] adopted the MapReduce programming model to realize the minimum and maximum modular SVM's SMO algorithm. The random gradient projection algorithm in the above algorithm is used to solve the objective function by random gradient descent; in each iteration, a training sample is randomly selected to calculate the gradient of the objective function, and then a step is preset in the opposite direction. The running time of the algorithm satisfies $O(d/\lambda\varepsilon)$, where d is the number of nonzero features on the boundary in each sample. The stochastic gradient descent algorithm has the lowest efficiency in many SVM serial solving algorithms, but can produce the best generalization performance. Shalev-Shwartz [17] pointed out that the number of iterations required for the stochastic gradient projection as the SVM solution method is $\tilde{O}(1/\varepsilon)$, where ε is the precision of the solution, and the algorithm execution time does not depend on the size of the sample set and the scale cluster system can achieve the acceleration of the algorithm.

The stochastic subgradient projection algorithm is a typical algorithm for solving convex quadratic programming problems, which has a strong representativeness when it is used for solving SVM algorithms. Based on the deep analysis of the SVM solution process and stochastic subgradient projection algorithm, a parallel SVM algorithm using the stochastic subgradient projection algorithm and considering the structure of sample data is designed in this paper, with the help of the MapReduce model on the Hadoop cloud computing platform. The algorithm is applied to deal with the big historical data produced in the process of blast furnace production in order to obtain the efficient dynamic prediction model of [Si] in molten iron.

2. Foundation

2.1. Hadoop Framework. Hadoop, one of the distributed system infrastructure frameworks, has the advantages of high reliability, high efficiency, and freedom of scalability. It is a software platform for distributed processing of big data [18–20]. The key content of the software platform are MapReduce and Hadoop Distributed File System; the former is a distributed computing technology which is primarily responsible for computing, and the latter is a file system which is primarily responsible for data distributed storage.

The core idea of MapReduce distributed computing technology [21, 22] is to decompose a big-scale dataset into several small datasets, and then the multiple subnodes managed by one master node operate on several small data sets, and the computer realizes the aggregation of each child node and obtains the final big-scale data set operation result. In the MapReduce cluster, the application for pre-execution is called "job," with a JobTracker. The work units "task" are decomposed from the job run on the corresponding compute nodes and have several JobTrackers, where JobTracker directs MapReduce work to build the link between the application program and Hadoop. JobTracker determines the files to be processed, responsible for assigning and monitoring the corresponding nodes for different tasks, and TaskTracker is responsible for independently executing the specific task and interacting with JobTracker. If JobTracker fails to receive the information submitted by TaskTracker on a timely basis, TaskTracker is determined to crash and the task is determined to be broken and assigned to other nodes for processing.

The MapReduce programming model takes the key-value pair as its input form, and its execution process can be seen as a key-value pair conversion to another batch-value pair output process [23]. The MapReduce framework decomposes the input data into segments according to certain rules and parses it into a batch of

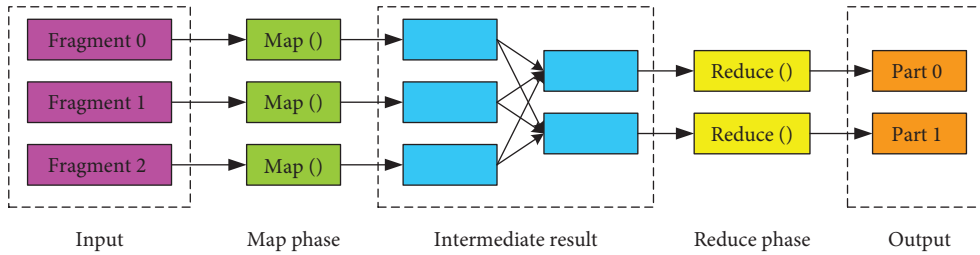

Figure 1: Process flow of MapReduce.

key-value pairs. During the Map phase, the user processes the key-value pair with the aid of the Map function to generate a series of intermediate key-value pairs during the Reduce phase. In the Reduce stage, the intermediate key-value pairs with the same key are aggregated to obtain the final result through the Reduce function. The specific process is shown in Figure 1. From Figure 1, we can see that users use the MapReduce programming model to write parallel programs and only need to use the Ctrip Map and Reduce functions of the program. Based on this, they do not need to consider the details of parallel execution of programs, thereby reducing the complexity of parallel programming and improving the ease of parallel programming.

HDFS is the foundation of data storage management in distributed computing, which has the advantages of high reliability, strong expansibility, and throughput. The premise and goal of the system design are as follows [24–27]:

(1) When HDFS is running on normal hardware, each component may be faulty, so error detection and quick recovery are the core goals of HDFS.

(2) The method of data reading on HDFS is massive data stream, so it requires high throughput when accessing data.

(3) Big-scale data sets require HDFS to support large file storage and can provide higher data transmission bandwidth.

(4) HDFS requires write once and read many times access model for documents, which simplifies data to meet high-throughput data access.

(5) The cost of mobile computing is lower than that of mobile data. A computing task is close to its operation data, which can reduce network congestion and improve system throughput.

(6) Heterogeneous hardware and software platforms should be portable.

HDFS uses master-slave architecture (see Figure 2); there is a NameNode and a plurality of DataNode. NameNode provides the ability to create, open, delete, rename, and direct the metadata of the file system. DataNode is used to store several pieces of data that are broken down, and each data block is copied into multiple copies and stored on different DataNode, so as to achieve the purpose of fault tolerance and disaster tolerance. By maintaining some data structures, NameNode records each information block divided by every file, as well as the status of each data node and other important information.

HSDS uses a file access model that writes one time but can be read multiple times, which makes HDFS data access with high throughput. In addition, HDFS also has a variety of reliability assurance measures, and it is stored in a number of copies of data to ensure data stability, also through data mode heartbeat detection, security mode, block reporting, and space recycling mechanism to ensure reliability.

2.2. Support Vector Machine. The support vector machine is a machine learning algorithm based on the VC dimension theory and structural risk minimization principle in statistical learning [28–30]. Mainly used in classification, according to the two classification problems, the main goal of this algorithm is to construct an optimal decision function ("best" based on the maximum interval theory), which can be used to classify the test data as accurately as possible. In the two classification problems of training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ (where $\mathbf{x}_i \in \mathbf{R}^n$ and $y_i \in \{+1, -1\}$), the SVM under linearly separable conditions, SVM under incomplete linearly separable conditions, and SVM under linearly separable conditions are discussed.

If there is a hyperplane which can correctly divide the sample sets of +1 and −1, the data set is called linear separable data set. The purpose of applying SVM is to find two hyperbranes with the largest class spacing based on the distance between the two classes. The hyperplane can be described by $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, where $\mathbf{w}$ is the normal vector for the class hyperplane and b is the offset. In Figure 3, the blue empty circle indicates category 1, the black empty circle represents category 2, the blue solid circle represents the boundary of category 1, and the black solid box represents the boundary of category 2; the two boundaries are called support vectors, and the midline of the vector (black solid line) indicates the desired optimal hyperplane.

Based on the idea of the maximum interval theory, the SVM model can be described by Formula (1). This formula describes the original problem of SVM, which is a typical convex quadratic programming problem with linear constraints. It can uniquely determine the maximum interval

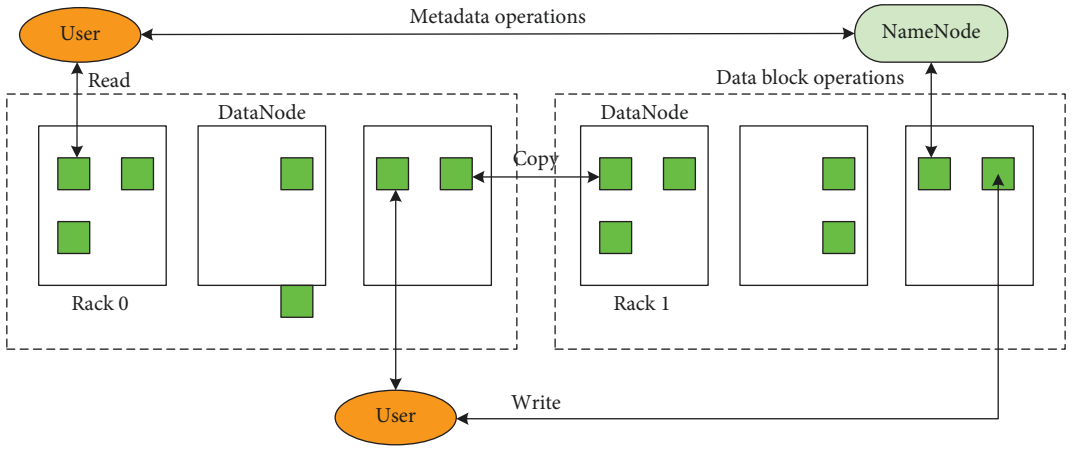

Figure 2: Structure sketch map of HDFS.

classification hyperplane, and its Lagrangian function can be described by (2).

$$\min_{\mathbf{w}} \quad \frac{\|\mathbf{w}\|^2}{2}, \qquad \text{s.t.} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \tag{1}$$

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i \langle \mathbf{w}, \mathbf{x}_i + b \rangle - 1), \tag{2}$$

In Formula (2), $\alpha_i \geq 0$ is the Lagrange multiplier corresponding to each sample, and the minimum value of $\mathbf{w}$ and b is obtained for the function $L(\mathbf{w}, b, \boldsymbol{\alpha})$; respectively, $\sum_{i=1}^{m} y_i \alpha_i = 0$ and $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ can be obtained according to the extreme conditions $\nabla_b L(\mathbf{w}, b, \boldsymbol{\alpha})$ and $\nabla_{\mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha})$, and bringing it into Formula (2), the dual problem of the primal problem of SVM can be obtained and can be described by

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \qquad \text{s.t.} \quad \sum_{i=1}^{m} y_i \alpha_i = 0, \quad \alpha_i \geq 0, \quad i = 1, 2, \ldots, m. \tag{3}$$

The optimal decision function of SVM can be described by Formula (4). $\mathbf{w}^*$ in Formula (4) is the optimal solution of (1), and $\boldsymbol{\alpha}^*$ is the optimal solution of (3).

$$f(\mathbf{x}) = \operatorname{sgn}\left(\langle \mathbf{w}^*, \mathbf{x} \rangle + b^*\right) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i^* y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b^*\right). \tag{4}$$

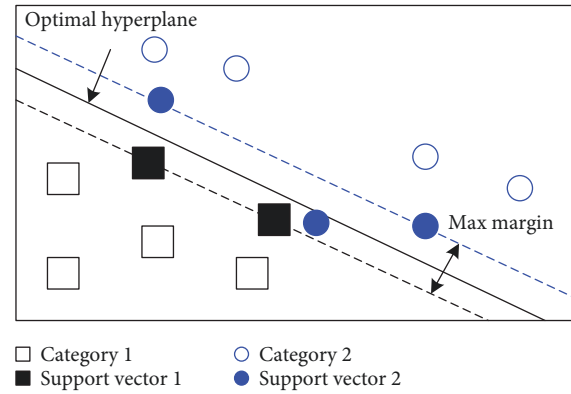

Figure 3: Linearly separable case of an SVM optimal hyperplane.

The sample set may have noise or other factors in the acquisition process, resulting in the sample set category being unable to be completely divided by a hyperplane [31, 32], but the vast majority of the sample points can be correctly divided by the hyperplane. Based on this, the relaxation variable is introduced for the purpose of constructing the optimal hyperplane with only a few samples indistinguishable. The relaxation variable is composed of the product relaxation variable ξ_i ($\xi_i \geq 0$) and C, so the original problem (1) becomes

$$\min_{\mathbf{w}} \quad \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{m} \xi_i, \qquad \text{s.t.} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, \ldots, m. \tag{5}$$

The purpose of introducing the penalty item is to enhance the fault tolerance of the SVM classifier. The larger the penalty factor is, the greater the penalty is. Equation (5), using hinge loss (or 0–1 loss function, or quadratic loss function, or other loss function), is not completely linearly

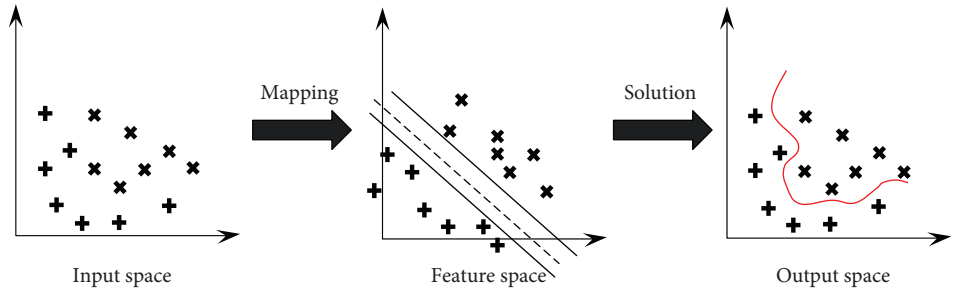

Figure 4: Nonlinear case of a schematic diagram of the SVM kernel function.

separable in the other form of (6). It is easy to get the dual problem of (5), using hinge loss (or 0–1 loss function, or quadratic loss function, or other loss function), which is another form of (6) in the condition of not being completely linearly separable.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{m} l(\mathbf{w}; (\mathbf{x}_i, y_i)), \\ \text{s.t.} \quad & l(\mathbf{w}(\mathbf{x}_i, y_i)) = \max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i\rangle\}, \end{aligned} \tag{6}$$

If the data sample set cannot be linearly separable, the original sample vector can be mapped from the original space to the higher-dimension feature space with a nonlinear function as shown in Figure 4, so that the sample can be linearly separable in the feature space. And then the optimal hyperplane is solved according to the maximum interval theory in the feature space. However, the complexity of algorithm calculations is increased in the process of solving the optimal hyperplane in high-dimensional space, which sometimes falls into a dimensional disaster. According to the Hilbert-Schmidt principle pointed out by the statistical learning theory, it is unnecessary to know the specific form of φ through the application of kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j)\rangle$. Since φ's kernel function is easier to obtain than φ itself, $\varphi(\mathbf{x}_i)$ will not be in the concrete solution appearing alone, but always appearing in the form of $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_i)$. At this time, the original problem of SVM can be described by (7), and the corresponding dual problem can be described by (8).

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{\|\mathbf{w}\|^2}{2}, \\ \text{s.t.} \quad & y_i(\langle \mathbf{w}, \varphi(\mathbf{x}_i)\rangle + b) \geq 1, \end{aligned} \tag{7}$$

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j)\rangle, \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{8}$$

3. SVM Solution Algorithm Based on Pegasos

The problem of solving SVM is essentially a quadratic programming problem. Related scholars had done a lot of research on the quadratic programming problem. Among them, the most typical ways are the Internal point method and the decomposition method.

The Internal point method [33]: the main idea of the former is using the penalty function instead of the inequality constraint and then using the iterative Newton method to optimize the solution. It has to solve twice of the SVM dual problem and get a number of optimization variables. For them, the Newton method is used to synchronize each variable step by step to optimize the synchronization. However, the time complexity of the interior point method for solving SVM problems is $O(n^3)$, and the spatial complexity is $O(n^2)$. Therefore, in the case of a large number of training samples, the use of an internal point method to solve an SVM problem is very difficult. Moreover, the use of an internal point method to obtain high accuracy does not mean that the generalization accuracy can also be improved.

The decomposition method [34, 35]: sequence minimal optimization is an effective method proposed by Platt in the 1980s to solve the SVM problem. Keerthi had improved the SMO algorithm and made a detailed theoretical proof of the convergence of the SMO algorithm. The improved SMO algorithm decomposed the quadratic programming problem of SVM into a series of subproblems. Based on the constraint of $\mathbf{y}^T\boldsymbol{\alpha} = 0$, the scale of the working sample set was minimized, and each of the two Lagrange multipliers which did not satisfy the KKT condition was optimized for each heuristic. Fix other parameters and use a simple analytical method to obtain the optimal solution of these two parameters, then update. Repeat the above process until all Lagrange multipliers meet the conditions. Although the decomposition method can effectively overcome the problem that the internal point of memory is too large, it must be transformed into a solution to its dual problem which will lead to its convergence rate which is significantly slower than the original problem of the direct solution.

The Pegasos is a gradient-based algorithm that can be applied to the direct solution to the original problem of SVM. The random gradient descent and the projection step are alternately performed in the iteration, and a number of samples are taken from the whole training sample to calculate the subgradient of each round.

Input: Training data set S, Iterations T, The number of k samples selected per round of iterations;
Output: The normal vector of a hyperplane w.
Main procedure:
1. Initialization vector w, arbitrarily select a vector w_1, and request $\|\mathbf{w}_1\| \le 1/\sqrt{\lambda}$;
2. For $t = 1 : T$
2.1 Select k samples from the training set S, subset $A_t \in S$, and replace the objective function with:

$$\min_{\mathbf{w}} \quad \lambda\|\mathbf{w}\|^2/2 + \sum_{(\mathbf{x},y)\in A_t} \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\}/k$$

2.2 Determine the learning efficiency $\eta_t = 1/\lambda t$ of the gradient descent method;
2.3 The use of w_t in A_t to determine the current loss of nonzero samples into a new subset

$$A_t^+ = \{(\mathbf{x}_i, y_i) \in A_t : y_i\langle \mathbf{w}_t, \mathbf{x}_i\rangle < 1\}$$

The sub-gradient direction of the objective function can be expressed as:

$$\nabla_t = \lambda \mathbf{w}_t - \frac{1}{|A_t|} \sum_{(\mathbf{x}_i, y_i)\in A_t^+} y_i \mathbf{x}_i$$

2.4 Update:

$$\mathbf{w}_{t+1/2} = \mathbf{w}_t - \eta_t \nabla_t = (1 - \eta_t \lambda)\mathbf{w}_t + \frac{\eta_t}{k} \sum_{(\mathbf{x}_i, y_i)\in A_t^+} y_i \mathbf{x}_i$$

2.5 Projection steps:

$$\mathbf{w}_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+1/2}\|}\right\}$$

3. Get the final result w_{T+1}.

Algorithm 1: Pegasos algorithm for solving SVM.

The SVM problem is described as

$$\min_{\mathbf{w}} \quad \frac{\lambda\|\mathbf{w}\|^2}{2} + \sum_{(\mathbf{x},y)\in S} \frac{\ell(\mathbf{w}; (\mathbf{x}, y))}{m}. \tag{9}$$

In formula (9),

$$\ell(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\}. \tag{10}$$

Formula (9) is expressed as $f(\mathbf{w})$, if

$$f(\widehat{\mathbf{w}}) \le \min_{\mathbf{w}} f(\mathbf{w}) + \varepsilon. \tag{11}$$

So $\widehat{\mathbf{w}}$ is the accuracy of the range of the ε solution.

The Pegasos requires that the number of the iterations is $\tilde{O}(1/\varepsilon)$, where ε represents the accuracy of the solution obtained. It can be seen that the execution time of the algorithm does not directly depend on the size of the sample set and should be applied to the big-scale data set classification problem.

The Pegasos algorithm is described as follows:

4. Structure and Parallelization of the Pegasos Algorithm Based on AP Clustering

The convergence of the Pegasos algorithm based on the SVM solution does not depend on the number of samples k which is selected for each iteration, but is constrained by the number of iterations T. When the products of k and T are fixed at an appropriate value, the two parameters have a certain interval, and the variation in the interval does not affect the convergence of the algorithm. The number of samples selected in each iteration of this study is $k = 1, 2, 3, \ldots, 10$, and the maximum number of iterations is $T = 10^5$, therefore $k \times T$ is between 10^5 and 10^6. Therefore, in dealing with big-scale samples and in each round of the iterative process, usually by selecting as many samples as possible reduces the number of iterations required for the algorithm. According to the above contents, the key steps of parallelizing the Pegasos algorithm are as follows [36–38]: randomly select the sample to obtain the objective function of the gradient direction of the process and through MapReduce break down to multiple machines in parallel. Each step of the gradient projection iteration is the same as 2.1–2.5 in Algorithm 1. But the parallel approach does not take into account the structural information of the sample, in order to effectively use the sample structure information and obtain a more reasonable SVM optimal superplanar classification. Based on the information of the sample structure, this paper proposed a Pegasos algorithm for parallel structured SVM.

The covariance matrix of the data contains the trend of data generation in the statistical sense, which can effectively reflect the result information of the data. The clustering information of data is important information for the structure of response data. This page uses the affinity propagation clustering algorithm [39], to provide a criterion for the sample data structure information and speed up the processing of big data samples; the parallel algorithm of the AP algorithm was designed.

4.1. AP Clustering Algorithm and Deserialization. The clustering mechanism of the AP clustering algorithm is "message passing." By passing messages between data points, the final cluster center can be identified. Involving two important parameters, the degree of attractiveness parameters and attribution parameters, respectively, the larger the former, the greater the likelihood that the candidate cluster center becomes the center of the actual cluster. The larger the latter, the greater the likelihood that the

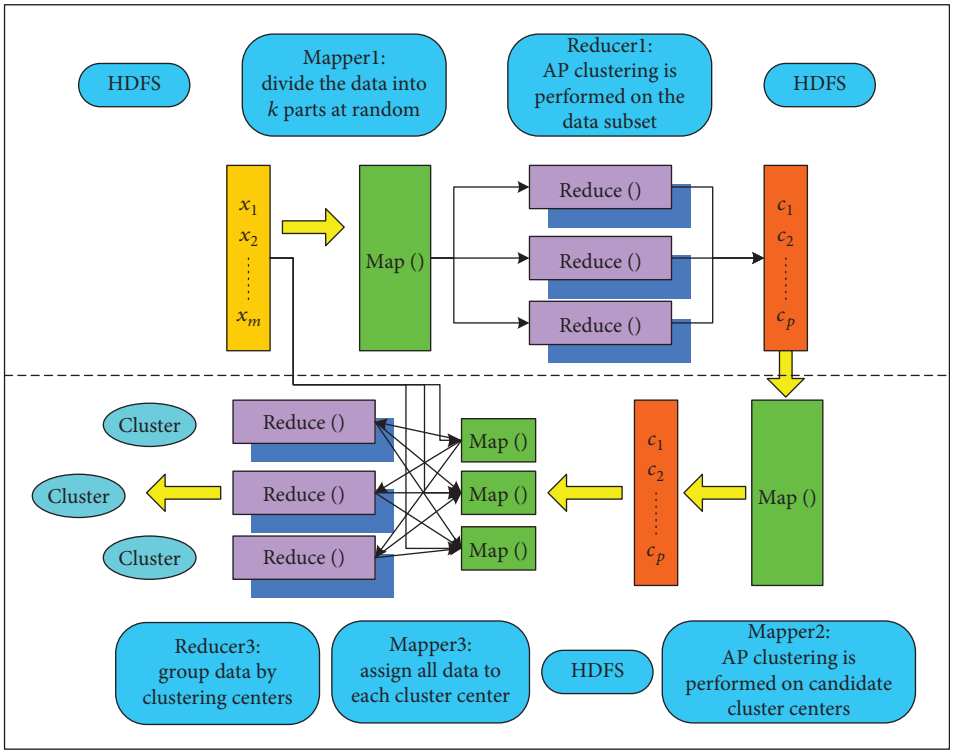

Figure 5: Implementation of distributed AP clustering based on MapReduce.

sample point belongs to the cluster center. The actual message passing mechanism is implemented by iteratively updating the attractiveness matrix $R = [r(i,k)]$ and the attribution degree matrix $A = [a(i,k)]$. The specific update rules are as follows [40–42]:

(1) The attribution matrix **R** is updated according to the membership degree matrix **A** and the similarity matrix $\mathbf{S} = [s(i,k)]$:

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \left\{ a\left(i,k'\right) + s\left(i,k'\right) \right\}. \tag{12}$$

(2) The membership matrix **A** according to the attraction degree matrix **R** and the similarity degree matrix **S**:

$$a(i,k) \leftarrow \min \left\{ 0, r(k,k) + \sum_{i' \neq i,k} \max\left\{0, r\left(i',k\right)\right\} \right\}, \tag{13}$$

$$a(k,k) \leftarrow \sum_{i' \neq k} \max\left\{0, r\left(i',k\right)\right\}. \tag{14}$$

In Formulas (12), (13), and (14), $s(i,k)$ represents the degree of similarity between point i and point k; the larger the value, the more likely it is that the k point is as an actual center, as it is the main basis for priority; initialization assigns the same value to all points, denoted as parameter p; $r(i,k)$ represents the degree of attraction of point i to point k; and $a(i,k)$ represents the degree of attribution of point i to point k. For point i, so that $a(i,k) + r(i,k)$ reaches the maximum, point k is their center; if $i = k$, then point k itself is a clustering center.

According to the basic idea of the AP clustering algorithm, in order to be able to effectively deal with big-scale data clustering problems, this paper designs a distributed AP based on MapReduce. For the implementation of the algorithm (see Figure 5), the process is divided into three phases: the first stage is Mapper1 and Reducer1, the second stage corresponds to Mapper2, and the third stage corresponds to Mapper3 and Reducer3. The algorithm is based on the MapReduce model, using AP clustering to sparse data, and then clustering represents the AP cluster again; this design can guarantee the clustering effect of the original AP algorithm but also improve the AP clustering efficiency and adaptability to different sizes of data.

4.2. Pegasos Algorithm of Structured SVM. The model of structured SVM can be described by

$$\min_{\mathbf{w}} \frac{\lambda_1}{2} \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{(\mathbf{x},y) \in S} \ell(\mathbf{w}; (\mathbf{x}, y)) + \frac{\lambda_2}{2} \mathbf{w}^T \sum \mathbf{w}. \tag{15}$$

In formula (15), $\ell(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x} \rangle\}$ is the loss function; λ_1 and $\lambda_2 > 0$ are the regularization factors to balance the options.

In the Pegasos algorithm, $\mathbf{w}$ is projected into aggregate $B = \{\mathbf{w} : \|\mathbf{w}\| \le 1/\sqrt{\lambda}\}$ after each iteration, but the structural information embedded in the Pegasos algorithm will cause the $\mathbf{w}$ projection range to change, so that the original projection range is no longer applicable. Therefore, the optimal constraints of the structured SVM will also change, which is expressed in (16). In Formula (16), $\sum$ is the covariance matrix of the sample, and the corresponding structural information is different from that of $\sum$, and $\mathbf{I}$ is the unit matrix.

$$\mathbf{w}^T\left(\lambda_1\mathbf{I} + \lambda_2\sum\right)\mathbf{w} \le 1. \tag{16}$$

If the optimal solution of (15) is $\mathbf{w}^*$, the constraints on the left side of the constraint condition described in (16) can be transformed into (17); the dual problem of the optimization problem described by (15) can be expressed by (18), and the optimal solution of the dual problem is $\boldsymbol{\alpha}^*$. From the strong duality theorem, we can see that the objective function value of the original problem is equal to the objective function value of the dual problem. In addition, (19) can be obtained by (17) and then simplified to obtain (20).

$$\left(\lambda_1\mathbf{I} + \lambda_2\sum\right)\mathbf{w}^* = \sum_{i=1}^{m}\boldsymbol{\alpha}_i^* y_i\mathbf{x}_i, \tag{17}$$

$$\max_{\boldsymbol{\alpha}\in[0,1/m]^m}\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j\mathbf{x}_i^T\left(\lambda_1\mathbf{I} + \lambda_2\sum\right)^{-1}\mathbf{x}_j, \tag{18}$$

$$\begin{aligned}\frac{\lambda_1}{2}\|\mathbf{w}^*\|^2 + \frac{1}{m}\sum_{i=1}^{m}\max\{0, 1 - y_i\langle\mathbf{w}^*, \mathbf{w}_i\rangle\} + \frac{\lambda_2}{2}\mathbf{w}^{*T}\sum\mathbf{w}^* \\ = \sum_{i=1}^{m}\alpha_i^* - \frac{1}{2}\mathbf{w}^{*T}\left(\lambda_1\mathbf{I} + \lambda_2\sum\right)\mathbf{w}^*,\end{aligned} \tag{19}$$

$$\mathbf{w}^{*T}\left(\lambda_1\mathbf{I} + \lambda_2\sum\right)\mathbf{w}^* = \sum_{i=1}^{m}\alpha_i^* - \frac{1}{m}\sum_{i=1}^{m}\max\{0, 1 - y_i\langle\mathbf{w}^*, \mathbf{w}_i\rangle\}. \tag{20}$$

The above analysis shows that the data structural information is embedded into the Pegasos algorithm, and the optimal solution of the structured Pegasos algorithm is found in set B (see Formula (21)). Therefore, after completing each iteration of the gradient descent program, $\mathbf{w}$ is projected into B, that is, the normal vector $\mathbf{w}$ is multiplied by Formula (22). This step can make $\mathbf{w}$ closer approximation to the optimal solution.

$$B = \left\{\mathbf{w} : \mathbf{w}^T\left(\lambda_1\mathbf{I} + \lambda_2\sum\right)\mathbf{w} \le 1\right\}, \tag{21}$$

$$\min\left\{1, \frac{1}{\sqrt{\mathbf{w}^T(\lambda_1\mathbf{I} + \lambda_2\sum)\mathbf{w}}}\right\}. \tag{22}$$

Algorithm 2 embeds the structure information into the initial objective function, which is characterized by the covariance matrix of the data samples, and the improvement of SVM original model is realized. The improved SVM algorithm achieves the time-consuming reduction of algorithm operation under the premise of satisfying the accuracy of algorithm classification. It is worth noting that the range of the descending sub-gradient direction and the range of the optimal solution of the Pegasos algorithm after embedding the structural information will change, but the intra-range fluctuation does not affect the application effect of SVM.

The Pegasos algorithm for structured SVM is described as follows:

4.3. Structure and Parallelization of the Pegasos Algorithm. Based on the Pegasos algorithm of structured SVM, the parallel processing of the structured Pegasos algorithm is implemented on the Hadoop platform by means of a MapReduce parallel framework model. The algorithm is divided into two stages which are the covariance parallel calculation phase of the data samples and the subgradient projection iterative parallel phase of the data samples. There is a separate MapReduce task for each iteration in the two stages of the calculation process.

When the sample data structural information is obtained under the MapReduce framework model, the training samples must be scattered on the corresponding data nodes to solve the covariance matrix of the data samples scattered on the corresponding data nodes. The training set S is divided into N subsets, which is denoted by (23), and the variance sum and the mean of the corresponding data subsets are denoted by (24).

$$S_i = \left\{\left(\mathbf{x}_j, y_j\right)\right\}_{j=1}^{N_i}, \quad i = 1, \ldots, N, y_j \in \{+1, -1\}, \tag{23}$$

$$\begin{aligned}\sum\nolimits_i &= \sum_{\mathbf{x}_j\in S_i}\mathbf{x}_j\mathbf{x}_j^T, \\ \mu_i &= \sum_{\mathbf{x}_j\in S_i}\mathbf{x}_j, \quad i = 1, \ldots, N.\end{aligned} \tag{24}$$

The covariance matrix on the training set S can be expressed as

$$\begin{aligned}\sum &= \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T, \\ \mu &= \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i.\end{aligned} \tag{25}$$

The simplified form of formula (25) available is

$$\begin{aligned}\sum &= \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i{\mathbf{x}_i}^T - \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i\frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_j \\ &= \frac{1}{m}\sum_{i=1}^{N}\sum\nolimits_i - \frac{1}{m^2}\sum_{i=1}^{N}\mu_i\sum_{i=1}^{N}\mu_i^T.\end{aligned} \tag{26}$$

Input: Training data set S, the number of iterations T, the number of samples k for each iteration;
Output: The normal vector of the classification hyperplane is w.
Main procedure:
1. Calculate the covariance matrix $\sum$ of the sample;
2. Initialize vector w Arbitrarily select a vector w_1 and ask for $\mathbf{w}^T(\lambda_1\mathbf{I} + \lambda_2\sum)\mathbf{w} \le 1$;
3. For $t = 1 : T$
3.1 Select the subset of the k samples, $A_t \in S$, from the training set S and replace $\min_{\mathbf{w}} \frac{\lambda_1}{2}\|\mathbf{w}\|^2 + \frac{1}{k}\sum_{(\mathbf{x},y)\in A_t} \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\} + \frac{\lambda_2}{2}\mathbf{w}^T\sum\mathbf{w}$ with the original objective function;
3.2 Determine the learning efficiency of the gradient descent method $\eta_t = 1/\lambda_1 t$;
3.3 A_t will use w_t to determine the current loss of nonzero samples into a new subset
$$A_t^+ = \{(\mathbf{x}_i, y_i) \in A_t : y_i\langle \mathbf{w}_t, \mathbf{x}_i\rangle < 1\}$$
The sub-gradient direction of the objective function can be expressed as:
$$\nabla_t = \lambda\mathbf{w}_t - \frac{1}{|A_t|}\sum_{(\mathbf{x}_i,y_i)\in A_t^+} y_i\mathbf{x}_i + \lambda_2\sum\mathbf{w}_t$$
2.4 Update:
$$\mathbf{w}_{t+1/2} = \mathbf{w}_t - \eta_t\nabla_t = (1 - \eta_t\lambda_1)\mathbf{w}_t + \frac{\eta_t}{k}\sum_{(\mathbf{x}_i,y_i)\in A_t^+} y_i\mathbf{x}_i - \eta_t\lambda_1\sum\mathbf{w}_t$$
2.5 Projection steps:
$$\mathbf{w}_{t+1} = \min\left\{1, 1/\sqrt{\mathbf{w}_{t+1/2}{}^T(\lambda_1\mathbf{I} + \lambda_1\sum)\mathbf{w}_{t+1/2}}\right\}\mathbf{w}_{t+1/2}$$
4. Get the final result w_{T+1}.

Algorithm 2: Pegasos algorithm for structured SVM.

With the aid of Formula (26) and the MapReduce model, the solution of the covariance matrix of the whole training set is solved. The specific algorithm is described as follows:

In conjunction with Algorithm 3, the algorithm of sub-gradient projection iteration is designed in parallel, and each iteration is taken as a single MapReduce task. The concrete steps for the design of the Map and Reduce algorithms for the T times iteration are as follows:

5. Algorithm of the Model and Example Analysis

5.1. Case Analysis and Data Collection. In this study, the deep excavation of big-scale historical production data of the No. 1 blast furnace (3200 m^3) of Tangshan Iron and Steel Company in the period from December 1, 2015, to November 30, 2016, was carried out to construct the [Si] content of molten iron during the smelting process for the prediction model.

(1) Sample output variable data collection: a total of 26 thousand samples of hot metal samples were collected during the iron cycle. To build a dynamic prediction model for the [Si] content of molten iron in the smelting process, the [Si] content of the 26 thousand molten iron collected was calculated according to the time series.

(2) Sample input variable data collection: the factors affecting the [Si] content of molten iron in the tap hole are sample input variables. Through the reaction mechanism and control mechanism in the process of blast furnace iron-making, the action diagram of the factors affecting the [Si] content of molten iron is shown in Figure 6. Twenty-four influencing factors X1~X24 were extracted as the sample input index [43–45] which can be obtained in real time. According to the frequency collected every 0.5 s, we can get 62.4 million sets of sample input data set.

5.2. Sample Set Construction. In this study, the deep excavation of big-scale historical production data of the No. 1 blast furnace (3200 m^3) of Tangshan Iron and Steel Company in the period from December 1, 2015, to November 30, 2016, was carried out to construct the [Si] content of molten iron during the smelting process for the prediction model. [Si] content means mass fraction.

The sample set is composed of sample input and sample output. Based on the reaction mechanism and control mechanism in the blast furnace iron-making process, the 24 indicator factors shown in Figure 6 are extracted as sample input. Using the gray relational model shown in (27) and (28), the correlation between the 24 input indices and the output [Si]% is calculated; that is, sample input index selection depends on the size of the correlation degree. Combined with the correlation degree calculation result and the blast furnace smelting process control principle, the correlation degree threshold is determined to be 0.870. The gray correlation between the 24 calculated input parameters and [Si]% is shown in Table 1. As can be seen from Table 1, excluding the gray correlation degree is less than the threshold of alternative indicators, to retain more than the threshold of alternative indicators.

$$\xi_i(k) = \frac{\Delta_{\min} + \rho\Delta_{\max}}{|y_0(k) - y_i(k)| + \rho\Delta_{\max}}, \tag{27}$$

$$\gamma(x_0, x_i) = \sum_{k=1}^{n}\frac{\xi_i(k)}{n}. \tag{28}$$

//Map
Input: The sample on the current node;
Output: N_i, $\sum_i$, μ_i.
Main procedure:
1. Scan the current node sample, accumulate the number of samples of the current node N_i;
2. Calculate $\sum_i \sum_{x_j \in S_i} \mathbf{x}_j \mathbf{x}_j^T$;
3. Calculate $\mu = \sum_{x_j \in S_i} \mathbf{x}_j$;
//Reduce
Input: Each node N_i, $\sum_i$, μ_i. $i = 1, \ldots, N$;
Output: $\sum_A$.
Main procedure:
1. Summarize the output of the Map node;
2. Find the covariance matrix $\sum_A$ of the whole sample according to Eq. (26).

Algorithm 3: Parallel design of sample covariance matrix for training set.

//Map
Input: w_t, The number of nodes M, the number of random samples taken per iteration k;
Output: The current node obtained $v_j = \sum_{(\mathbf{x}_i, y_i) \in A_t^+} y_i \mathbf{x}_i$, among them: $A_t^+ = \{(x_i, y_i) \in A_t : y_i \langle w_t, x_i \rangle < 1\}$.
Main procedure:
1. Randomly selected k/M samples;
2. Define the zero vector $v_j \in R^n$;
3. For $i = 1$ to k/M
If $y_i^* = y_i{}^* \mathbf{w}_t^T \mathbf{x}_i < 1$
Then $v_j = v_j + y_i^* x_i$
4. Solving get $v_{j.}$
//Reduce
Input: The current number of iterations t, w_t, the number of nodes M, the number of random samples taken per iteration k;
Output: w_{t+1}.
Main procedure:
1. Calculate $\mathbf{v} = \sum_{j=1}^{M} \mathbf{v}_j$;
2. Calculate $\eta_t = \lambda_1{}^* t$;
3. Calculate:
$\mathbf{w}_{t+1/2} = (1 - \eta_t \lambda_1)\mathbf{w}_t + \frac{\eta_t}{k}\mathbf{v} - \eta_t \lambda_2 \sum \mathbf{w}_t$;
4. Calculate:

$$\mathbf{w}_{t+1} = \min\left\{1, 1/\sqrt{\mathbf{w}_{t+1/2}{}^T (\lambda_1 I + \lambda_2 \sum) \mathbf{w}_{t+1/2}}\right\} \mathbf{w}_{t+1/2}$$

Algorithm 4: Parallel algorithm design of Pegasos' tth iteration.

In formula (27), $\xi_i(k)$ is the correlation coefficient, ρ is the differential coefficient, satisfying $\rho \in [0, 1]$, $\Delta_{\min} = \min_i (\min_k |y_0(k) - y_i(k)|)$, and $\Delta_{\max} = \max_i (\max_k |y_0(k) - y_i(k)|)$.

In Formula (28), the gray relation degree is denoted by $\gamma(x_0, x_i)$.

The blast furnace operation process is a large delay process, corresponding to the time synchronization which corresponds to the input index and [Si]% data between the existence of a large delay. In order to improve the accuracy of the prediction model, it is necessary to modify the order of the delay between the input and the output to construct the most relevant sample set. In this study, the correlation coefficient analysis method is used to calculate the influence degree of input variables on [Si]% at different time delays. Among them, the time series is set to 0T, 1T, 2T, 3T, 4T, and 5T in terms of the hot metal cycle. The smelting period T of the No. 1 (3200 m^3) blast furnace in Tangshan Iron and Steel II Factory is 6 ~ 8 hours. The correlation coefficient calculation results are shown in Table 2.

62.4 million sample input datasets were integrated according to the smelting period and one-to-one correspondence with the content of [Si] in molten iron; after that, a sample set size of 26,000 was formed. Then, extract x1 ~ x19 as sample input, and design the essence sample set (sample size is 25996) corresponding to the sample input and

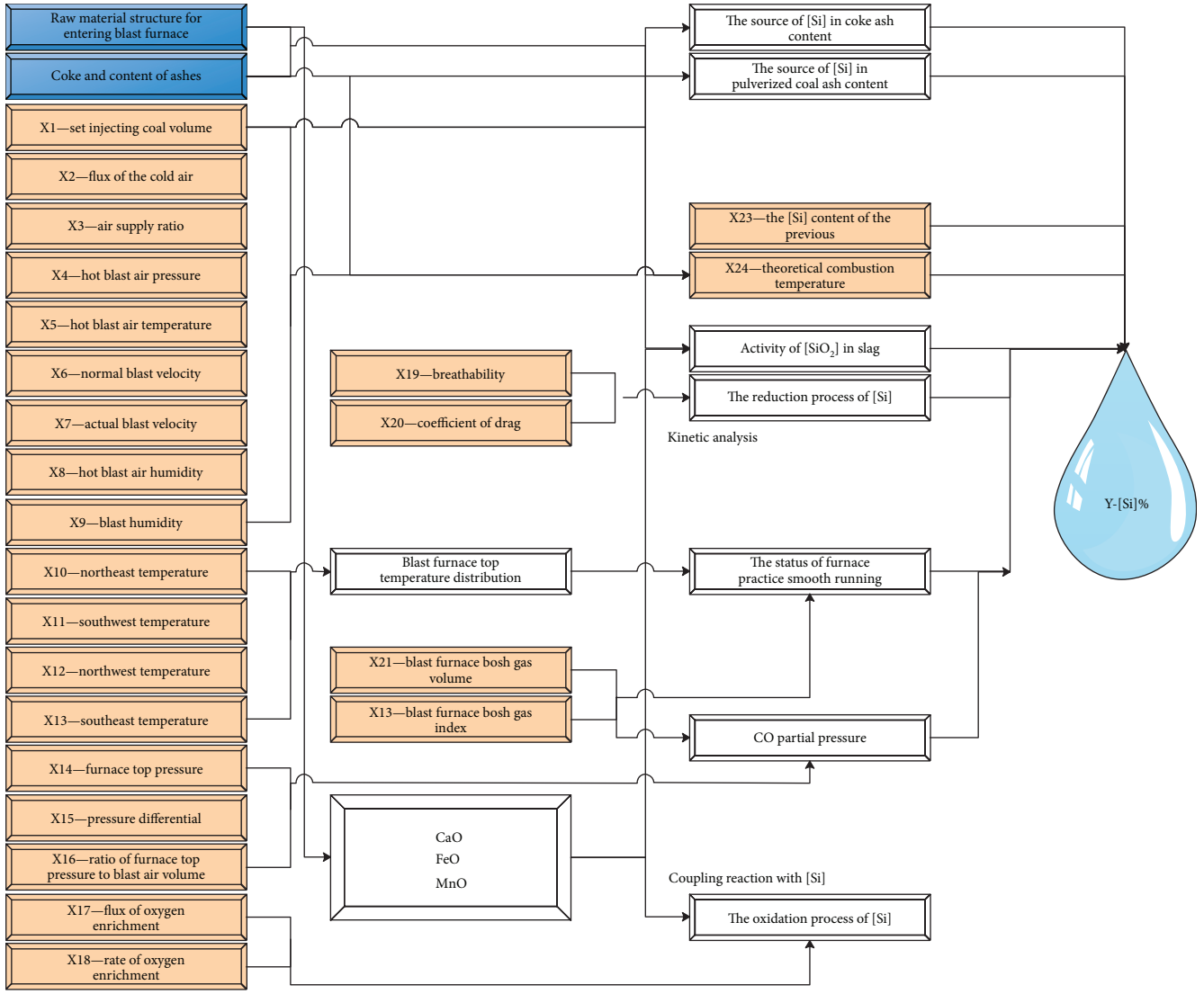

Figure 6: Action diagram of influencing factors of [Si] content in molten iron.

output according to the correlation statistical analysis results in Table 2. The correspondence of the essence sample set is as follows:

$$f[x_1(k), x_2(k-3), x_3(k-3), x_4(k), x_5(k-3), x_6(k), x_7(k), x_8(k-1), x_9(k-1), x_{10}(k-1), x_{11}(k-1), x_{12}(k-3), x_{13}(k), x_{14}(k-2), x_{15}(k-2), x_{16}(k-3), x_{17}(k-3), x_{18}(k), x_{19}(k-1)] = [\mathrm{Si}](k+1). \tag{29}$$

5.3. Experimental Design

5.3.1. Sample Set. In this paper, the support vector machine parallel algorithm is designed, and a serum sample set is determined based on the [Si] content of the dynamic prediction model. Select the 24996 group of samples as the training set in chronological order, after selecting the group of samples as a test set of 1000.

5.3.2. Experimental Platform. Experiments were carried out on personal computers and cloud computing platforms. Personal computer configuration is 3.20 GHz frequency, with 8 GB memory, and cloud computing platform configuration is 1 master node server and 20 slave node servers. Each node processor is Intel® Xeon® CPU E5620, the frequency is 2.40 GHz, the operating system is 64-bit Debian Linux, and the Hadoop platform version is hadoop-0.20.2.

5.3.3. Parameter Setting. The number of iterations in the Pegasos algorithm is $T = 10^5$; the number of samples selected for each iteration is $k = 1$, $\lambda = 0.01$; the number of iterations in the parallel Pegasos algorithm is $T = 10^5$; the number of samples selected for each iteration is $k = 1$, $\lambda_1 = 0.001$, and $\lambda_2 = 0.01$; the structure of the Pegasos algorithm is applied to AP clustering; and the training sample set on AP clustering parameters is $p = -3$, then we can get four clusters.

Table 1: List of results of alternative input indicator and [Si]% gray relational degree solving.

Variable symbol	Gray correlation	Is larger than a threshold
X1	0.8702	✓
X2	0.8861	✓
X3	0.8860	✓
X4	0.8641	✗
X5	0.8737	✓
X6	0.8862	✓
X7	0.8772	✓
X8	0.8689	✗
X9	0.8864	✓
X10	0.8852	✓
X11	0.8860	✓
X12	0.8896	✓
X13	0.8811	✓
X14	0.8814	✓
X15	0.8666	✗
X16	0.8804	✓
X17	0.8199	✗
X18	0.8195	✗
X19	0.8842	✓
X20	0.8827	✓
X21	0.8848	✓
X22	0.8853	✓
X23	0.9160	✓
X24	0.8832	✓

5.3.4. Algorithm Evaluation. Algorithm performance indicators are divided into two categories; in the [Si] content dynamic prediction simulation, the predicted hit rate (HR) and the training sample set are used to study the learning time t, and the predicted hit rate HR is used in the [Si] content rise and fall prediction simulation, training sample learning time true positive rate TPR, true negative rate TNR, and geometric mean (GM). Among them, the predictive hit rate is calculated as (30), the true positive rate TPR is calculated as (31), the true negative rate of TNR is calculated as (32), and the geometric mean GM is calculated as (33).

$$\mathrm{HR} = \frac{n_{<|\varepsilon|}}{N} \times 100\%, \tag{30}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \times 100\%, \tag{31}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \times 100\%, \tag{32}$$

$$\mathrm{GM} = \sqrt{\mathrm{TPR} \times \mathrm{TNR}}. \tag{33}$$

Formulas (30), (31), (32), and (33): $n_{<|\varepsilon|}$ is the number of predicted times less than the error threshold in the [Si] content prediction process, N is the total number of samples, TP is the number of times that the [Si] content actually rises and is predicted to rise, FN is the number of times that the [Si] content actually rises but is predicted to decrease, FP is the number of times that the [Si] content actually drops but is forecast to rise, and TN is the number of times that the [Si] content is actually decreased and the prediction is decreased.

Table 2: Different time delay correlation coefficients under the input variables and the calculation results of the [Si]% list.

Input variable the previous n hours	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
x_1	**0.2875**	0.2360	0.2044	0.1751	0.1316	0.0897
x_2	0.0167	0.0078	0.0836	**0.1068**	0.0530	0.0559
x_3	0.0140	0.0099	0.0847	**0.1077**	0.0542	0.0554
x_4	**0.4804**	0.4362	0.3407	0.2581	0.2073	0.1532
x_5	0.0164	0.0077	0.0834	**0.1073**	0.0534	0.0578
x_6	**0.2781**	0.2652	0.2740	0.2415	0.1498	0.1204
x_7	**0.1250**	0.1190	0.1033	0.0994	0.0969	0.0764
x_8	0.2886	**0.3551**	0.3512	0.3058	0.3123	0.3282
x_9	0.3258	**0.3691**	0.3580	0.3139	0.3113	0.3077
x_{10}	0.3631	**0.4266**	0.4234	0.3810	0.3659	0.3596
x_{11}	0.3655	**0.4212**	0.4163	0.3661	0.3657	0.3634
x_{12}	0.0849	0.0562	0.0951	**0.0999**	0.0470	0.0420
x_{13}	**0.1386**	0.0239	0.0194	0.0324	0.0238	0.0255
x_{14}	0.0166	0.0626	**0.1350**	0.1315	0.0403	0.0418
x_{15}	0.0126	0.1462	**0.1520**	0.1105	0.1133	0.1081
x_{16}	0.0196	0.0499	0.1233	**0.1432**	0.0867	0.0903
x_{17}	0.0223	0.0532	0.1281	**0.1447**	0.0886	0.0906
x_{18}	**0.6827**	0.5360	0.4059	0.2698	0.1854	0.1124
x_{19}	0.2578	**0.2884**	0.2852	0.2570	0.1989	0.1940

For the sake of convenience, first renumber the index after the gray relational model [46] optimization: "X1 == x_1"; "X2 == x_2"; "X3 == x_3"; "X5 == x_4"; "X6 == x_5"; "X7 == x_6"; "X9 == x_7"; "X10 == x_8"; "X11 == x_9"; "X12 == x_{10}"; "X13 == x_{11}"; "X14 == x_{12}"; "X16 == x_{13}"; "X19 == x_{14}"; "X20 == x_{15}"; "X21 == x_{16}"; "X22 == x_{17}"; "X23 == x_{18}"; "X24 == x_{19}".

5.4. Dynamic Prediction of [Si] Content and Analysis of the Results. Based on the above experimental design criteria, the [Si] content of the blast furnace iron-making process is predicted, and the predicted hit rate of the three algorithms is shown in Table 3. It can be seen from Table 3 that the SVM algorithm solved by serial Pegasos shows the best effect in predicting the hit rate of the [Si] content, but the sample training time is significantly longer than the other two algorithms. The SVM algorithm with a structural parallel Pegasos solution has a lower hit rate than the serial algorithm, but the sample training time is significantly better than the serial algorithm. Only the parallel algorithm does not consider the sample data structure of the SVM algorithm which has the worst accuracy, but the sample training time is significantly better than for the serial algorithm, compared

Table 3: Statistical table of dynamic prediction of the [Si]% value of three different algorithms.

Error range	SVM algorithm for the serial Pegasos solver		The parallel SVM algorithm for Pegasos		SVM algorithm for the structural parallel Pegasos solver	
	Hit rate	The cumulative hit rate	Hit rate	The cumulative hit rate	Hit rate	The cumulative hit rate
$\|\varepsilon\| < 0.01$	33.2%	33.2%	30.1%	30.1%	32.9%	32.9%
$0.01 < \|\varepsilon\| < 0.02$	17.1%	50.3%	16.5%	46.6%	18.4%	51.3%
$0.02 < \|\varepsilon\| < 0.03$	18.4%	68.7%	11.7%	58.3%	14.2%	65.5%
$0.03 < \|\varepsilon\| < 0.04$	14.1%	82.8%	15.6%	73.9%	11.3%	76.8%
$0.04 < \|\varepsilon\| < 0.05$	2.5%	85.3%	10.4%	84.3%	7.4%	84.2%
$0.05 < \|\varepsilon\| < 0.06$	3.1%	88.4%	3.0%	87.3%	7.0%	91.2%
$\|\varepsilon\| < 0.06$	11.6%	100%	12.7%	100%	8.8%	100%
Time (ms)	105032926 (order of magnitude 10^9)		1921829 (order of magnitude 10^7)		1928397 (order of magnitude 10^7)	

Table 4: Summarization of statistical results of [Si]% lifting dynamic prediction performance of three different algorithms.

Error range	HR	TPR	TNR	GM	Time (ms)
Interpolation of SVM algorithm for serial Pegasos	77.50%	72.31%	77.90%	74.92%	9,004,328
Parallel Pegasos solving SVM algorithm	82.55%	63.13%	84.07%	72.62%	1,776,882
Parametric parallel Pegasos solving SVM algorithm	92.2%	85.47%	86.33%	85.90%	1,710,788

with the structural parallel Pegasos that solved SVM. The algorithm has roughly the same sample training time. Therefore, the structure of this study designed parallel Pegasos algorithm has a stronger practicality.

The predictive hit rate HR, the true positive rate TPR, the true negative rate TNR, and the geometric mean GM of the three algorithms are predicted in Table 4 of the blast furnace iron-making process. According to the results of the three algorithms presented in Table 4, we can see that the SVM algorithm, which is designed in this paper, has stronger practicability in the prediction of [Si] content lift.

The distribution of the results around the actual results in the blast furnace [Si] content prediction is shown in Figure 7. It can be seen from Figure 7 that the prediction results are mostly concentrated in the actual value of ±0.06 and in terms of accuracy are to meet the actual needs. Figure 8 shows the predictive results and actual results in the timing of the control situation; it can be achieved in real-time [Si] dynamic forecast for the actual production of the blast furnace to provide guidance.

Compared with the evaluation of the algorithm, the SVM algorithm of structure and parallelism not only preserves the excellent performance of SVM (the accuracy of the absolute error can reach 91.2% of the predicted hit rate), and the algorithm of SVM speed has improved significantly (time-consuming than the serial algorithm which increased by 54 times). In addition, a 92.2% high hit rate was obtained in the notice of dynamic fluctuation of molten iron [Si], and the SVM algorithm was improved by 5 times in the solution speed.

It is noteworthy that the silicon content prediction algorithm is an analog quantity prediction. It has high precision requirements for numerical values. The traditional serial

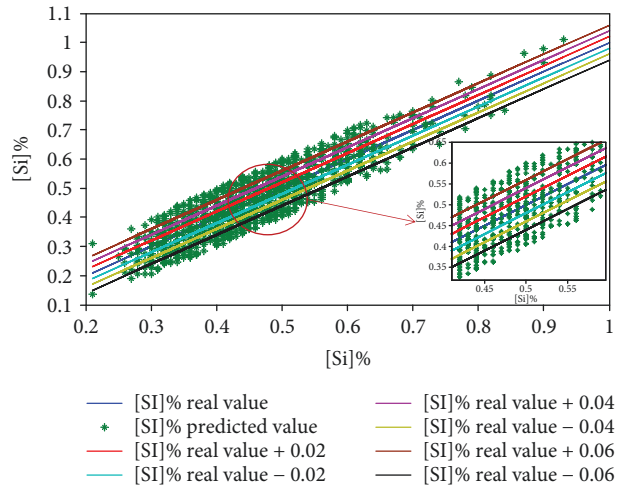

Figure 7: Fluctuation of dynamic loading results in different regions of true value.

algorithm takes a lot of time. The parallel structured SVM algorithm can improve the learning speed of the algorithm more effectively. The silicon content up-and-down prediction algorithm is a two-classification problem, which has low precision requirements for numerical values and focuses on pattern recognition, and the traditional serial algorithm takes less time. The effect of using the parallel structured SVM algorithm to improve the learning speed of the algorithm is not obvious. This research carried out scientific algorithm design and simulation experiments. The empirical results show that the time-consuming improvement effect of the silicon content prediction algorithm is

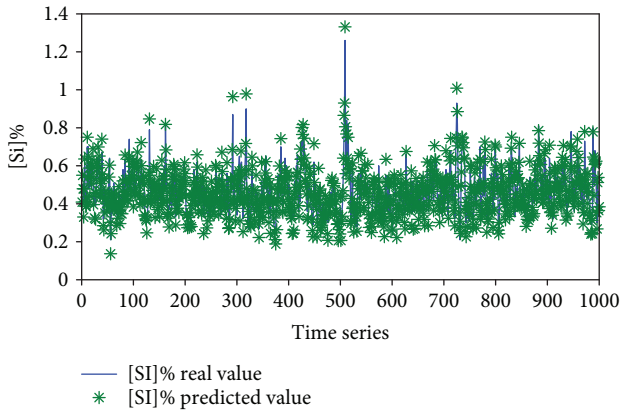

Figure 8: Comparison of dynamic prediction results and actual values of [Si] content.

significantly higher than the silicon content up-and-down prediction algorithm.

6. Conclusions

The support vector machine is a machine learning algorithm based on the maximum interval theory. The biggest advantage of this algorithm is that it can avoid dimensionality disaster with kernel function and realize the maximum generalization performance of the algorithm by means of the structural risk minimization principle, and the algorithm is mainly applicable to small sample data. However, the processing of big data samples is not optimistic, especially in the solution of SVM which presents a serious shortage. Therefore, it is very necessary to design a parallel SVM algorithm in the Hadoop platform to improve the algorithm to solve the speed. This study researches the SVM algorithm for stochastic subgradient projection, and based on the characteristics of the execution time of the algorithm, a stochastic subgradient algorithm based on AP clustering is designed and used in SVM solution.

In the algorithmic link, the data set of the size of $6240 \times 104 \times 37$ is processed, and the result of dynamic prediction of blast furnace [Si] content is obtained by using the random subgradient algorithm with fine sample set and application structure and parallelization.

In view of the advantages of the algorithm in forecasting accuracy and algorithm solving speed, it is worthy to be popularized in [Si] dynamic forecasting of blast furnace iron-making. The promotion route is divided into the following three steps:

Step 1. Based on the Hadoop platform, the SVM parallel algorithm is designed to train the sample data, and then the best [Si] content dynamic forecasting model is applied to the practice.

Step 2. Based on the practical effect of the target customers, the new data of the on-site production process is supplemented into the training samples to optimize the dynamic prediction model of the [Si] content.

Step 3. The SVM parallel algorithm is designed in this paper to configure the Hadoop platform for the blast furnace production site in real time. The real-time updating of the real-time data collection and training sample set is realized, and the [Si] dynamic forecasting model is provided in real time.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was financially supported in part by the National Natural Science Foundation of China (51504080), in part by the Science and Technology Project of Hebei Education Department (BJ2017021), and in part by the Outstanding Youth Fund Project of North China University of Science and Technology (JQ201705).

References

[1] S. Rosset, J. Zhu, and T. Hastie, "Boosting as a regularized path to a maximum margin classifier," *Journal of Machine Learning Research*, vol. 5, no. 4, pp. 941–973, 2004.

[2] V. Vapnik, "Principles of risk minimization for learning theory," in *Advances in Neural Information Processing Systems*, pp. 831–838, DBLP, 1992.

[3] B. Kaltenbacher, "A posteriori parameter choice strategies for some Newton type methods for the regularization of nonlinearill-posed problems," *Numerische Mathematik*, vol. 79, no. 4, pp. 501–528, 1998.

[4] K. Koh, S. J. Kim, and S. Boyd, "An interior-point method for big-scale l 1-regularized logistic regression," 2007, http://JMLR.org.

[5] L. W. Sun and J. Li, "BISVM: block-based incremental training algorithm of SVM for very big dataset," *Application Research of Computers*, vol. 25, no. 1, pp. 98–100, 2008.

[6] S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone, "A convergent hybrid decomposition algorithm model for SVM training," *IEEE Transactions on Neural Networks*, vol. 20, no. 6, pp. 1055–1060, 2009.

[7] S. S. Keerthi and E. G. Gilbert, "Convergence of a generalized SMO algorithm for SVM classifier design," *Machine Learning*, vol. 46, no. 1/3, pp. 351–360, 2002.

[8] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: distributed internet computing for IT and scientific research," *IEEE Internet Computing*, vol. 13, no. 5, pp. 10–13, 2009.

[9] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. E98.B, no. 1, pp. 190–200, 2015.

[10] N. Mustafee, "Exploiting grid computing, desktop grids and cloud computing for e-science: future directions," *Transforming Government: People, Process and Policy*, vol. 4, no. 4, pp. 288–298, 2010.

[11] K. Dhanya and S. Preethi, "A virtual cloud computing provider for mobile devices," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 3, no. 3, pp. 411–414, 2017.

[12] B. Liu, J. Chen, and S. Wang, "Protein remote homology detection by combining pseudo dimer composition with an ensemble learning method," *Current Proteomics*, vol. 13, no. 2, pp. 86–91, 2016.

[13] S. Xiaole, L. Jianhua, L. Rui, and L. Xia, "Paralleled optimal power flow algorithm based on auxiliary problem principle and interior point algorithm," *Journal of Xian Jiaotong University*, vol. 40, no. 4, p. 468, 2006.

[14] X. Feng, W. Hao, and T. Jianeng, "Compressed sensing reconstruction based on subgradient projection method," *Journal of Convergence Information Technology*, vol. 7, no. 6, pp. 9–15, 2012.

[15] J.-x. Dong, A. Krzyzak, and C. Y. Suen, "Fast SVM training algorithm with decomposition on very large data sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 603–618, 2005.

[16] J. Zhao, Z. Liang, and Y. Yang, "Parallelized incremental support vector machines based on MapReduce and Bagging technique," in *2012 IEEE International Conference on Information Science and Technology*, pp. 297–301, Hubei, China, 2012.

[17] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[18] A. Rabkin and R. H. Katz, "How Hadoop clusters break," *IEEE Software*, vol. 30, no. 4, pp. 88–94, 2013.

[19] L. Wang, J. Tao, R. Ranjan et al., "G-Hadoop: MapReduce across distributed data centers for data-intensive computing," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 739–750, 2013.

[20] M. Niemenmaa, A. Kallio, A. Schumacher, P. Klemelä, E. Korpelainen, and K. Heljanko, "Hadoop-BAM: directly manipulating next generation sequencing data in the cloud," *Bioinformatics*, vol. 28, no. 6, pp. 876-877, 2012.

[21] J. Dean and S. Ghemawat, "Map reduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, p. 107, 2008.

[22] R. Lämmel, "Google's MapReduce programming model — revisited," *Science of Computer Programming*, vol. 70, no. 1, pp. 1–30, 2008.

[23] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 922–933, 2009.

[24] X. Liu, J. Han, Y. Zhong, C. Han, and X. He, "Implementing WebGIS on Hadoop: a case study of improving small file I/O performance on HDFS," in *2009 IEEE International Conference on Cluster Computing and Workshops*, pp. 1–8, New Orleans, LA, USA, 2009.

[25] A. K. Karun and K. Chitharanjan, "A review on hadoop — HDFS infrastructure extensions," in *2013 IEEE Conference on Information and Communication Technologies*, pp. 132–137, Thuckalay, Tamil Nadu, India, 2013.

[26] A. Higai, A. Takefusa, H. Nakada, and M. Oguchi, "A study of effective replica reconstruction schemes for the Hadoop distributed file system," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 4, pp. 872–882, 2015.

[27] S. Park and Y. Lee, "Secure Hadoop with encrypted HDFS," in *Grid and Pervasive Computing*, pp. 134–141, Springer, Berlin Heidelberg, 2013.

[28] M. M. Adankon and M. Cheriet, "Support vector machine," in *Third International Conference on Intelligent Networks and Intelligent Systems IEEE Computer Society*, pp. 418–421, Chennai, India, 2010.

[29] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[30] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[31] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.

[32] A. Widodo and B. S. Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *MechanicalSystems and Signal Processing*, vol. 21, no. 6, pp. 2560–2574, 2007.

[33] T. Li, H. Li, X. Liu, S. Zhang, K. Wang, and Y. Yang, "GPU acceleration of interior point methods in large scale SVM training," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 863–870, Melbourne, VIC, Australia, 2013.

[34] B. H. Yang, D. Chen, D. X. Zheng, and D. F. Zha, "Fault diagnosis method for bearing based on wavelet packet decomposition and EMD-SVM," *Computer Measurement and Control*, vol. 23, no. 4, pp. 1118–1120, 2015.

[35] A. P. Dobrowolski, M. Wierzbowski, and K. Tomczykiewicz, "Multiresolution MUAPs decomposition and SVM-based analysis in the classification of neuromuscular disorders," *Computer Methods and Programs in Biomedicine*, vol. 107, no. 3, pp. 393–403, 2012.

[36] J. F. de Oliveira and M. S. Alencar, "Online learning early skip decision method for the HEVC inter process using the SVM-based Pegasos algorithm," *Electronics Letters*, vol. 52, no. 14, pp. 1227–1229, 2016.

[37] V. Jumutc, X. Huang, and J. A. K. Suykens, "Fixed-size Pegasos for hinge and pinball loss SVM," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, Dallas, TX, USA, 2013.

[38] Y. W. Chang, C. J. Hsieh, K. W. Chang, M. Ringgaard, and C. J. Lin, "Training and testing low-degree polynomial data mappings via linear SVM," *Journal of Machine Learning Research*, vol. 11, pp. 1471–1490, 2010.

[39] K. Wang, J. Zhang, D. Li, X. Zhang, and T. Guo, "Adaptive affinity propagation clustering," http://arxiv.org/abs/0805.1096.

[40] C. Shea, B. Hassanabadi, and S. Valaee, "Mobility-based clustering in VANETs using affinity propagation," in *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pp. 1–6, Honolulu, HI, USA, 2009.

[41] J. Vlasblom and S. J. Wodak, "Markov clustering versus affinity propagation for the partitioning of protein interaction graphs," *BMC Bioinformatics*, vol. 10, no. 1, p. 99, 2009.

[42] R. Guan, X. Shi, M. Marchese, C. Yang, and Y. Liang, "Text clustering with seeds affinity propagation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 627–637, 2011.

[43] N. Tsuchiya, M. Tokuda, and M. Ohtani, "The transfer of silicon from the gas phase to molten iron in the blast furnace," *Metallurgical Transactions B*, vol. 7, no. 3, pp. 315–320, 1976.

[44] T. Bhattacharya, "Prediction of silicon content in blast furnace hot metal using partial least squares (PLS)," *ISIJ International*, vol. 45, no. 12, pp. 1943–1945, 2005.

[45] L. Jian, C. Gao, L. Li, and J. Zeng, "Application of least squares support vector machines to predict the silicon content in blast furnace hot metal," *ISIJ International*, vol. 48, no. 11, pp. 1659–1661, 2008.

[46] H. Jiang and W. He, "Grey relational grade in local support vector regression for financial time series prediction," *Expert Systems with Applications*, vol. 39, no. 3, pp. 2256–2262, 2012.