Engineering Applications of Artificial Intelligence 92 (2020) 103643

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Distributed gas concentration prediction with intelligent edge devices in coal mine☆

Yiwen Zhang [a], Haishuai Guo [a], Zhihui Lu [b,c,*], Lu Zhan [d,e], Patrick C.K. Hung [f]

[a] Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei, China
[b] School of Computer Science, Fudan University, Shanghai, China
[c] Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, Shanghai, China
[d] School of Economics, Fudan University, Shanghai, China
[e] Shanghai Blockchain Engineering Research Center, Shanghai, China
[f] Faculty of Business and IT, University of Ontario Institute of Technology, Canada

ARTICLE INFO

Keywords:
Gas disaster
Intelligent edge system
Distributed gas concentration prediction
Industrial multidimensional data

ABSTRACT

Gas disaster can be triggered by gas concentrations exceeding standard levels, and gas concentration prediction system can reduce the occurrence of gas disaster by predicting the trend of gas concentration and alerting engineers to take necessary measures whenever needed. With the increasing use of intelligent edge devices in coal mines and the limitations of some existing systems, developing a new gas concentration prediction system for large-scale intelligent edge devices has become an important issue. This work proposes to address the issue through a novel method for predicting gas concentrations by taking full advantage of multidimensional data in an intelligent edge system. Specifically, 1) it proposed a Single hidden layer Random Weights Neural Network (SRWNN) as the prediction model, which is based on interval prediction rather than point prediction; 2) It employs a Non-dominated Sorting Genetic Algorithm II (NSGA-II) to train SRWNN; 3) To significantly reduce the time consumed during model training and facilitate real-time predictions, it proposes a distributed gas concentration prediction scheme based on an intelligent edge system; and 4) it conducts extensive experiments by using actual industrial data collected from a company to demonstrate the superior performance of the proposed method.

1. Introduction

Gas disaster is one of the main factors threatening the safety of coal mine production (Cheng et al., 2015), which often leads to considerable economic losses and casualties (Chao et al., 2015). Among all the gas disasters, gas concentration exceeding the limit is the main cause. Therefore, gas concentration prediction plays a vital role in ensuring the safety of coal mining production, and the predicted performance determines whether the coal mine can maintain high-level safety. Existing coal mine systems rely too much on the computing power of a computer center and fail to fully utilize the collected coal mine data, which has led engineers and researchers to explore new and effective prediction systems based on edge computing and machine learning.

In recent years, edge computing has become increasingly popular (Satyanarayanan, 2017; Chen et al., 2019; Wu et al., 2019; Xiao et al., 2019; Lu et al., 2018), which moves data processing, storage and service supply to intelligent edge devices. There are many application areas where edge computing can be used, such as an intelligent rare-event detection system (Janjua et al., 2019), smart city (Duan et al., 2019), and autonomous driving (Xiong et al., 2019). Nowadays, more and more coal mines choose to use intelligent edge devices to build intelligent edge systems. Traditional sensors can only collect data, while intelligent edge devices can perform partial calculations and storage, which mitigates the complex calculations in the center and reduces the traffic burdens of a network. However, most existing gas concentration prediction systems are difficult to transplant into a new system composed of intelligent edge devices, and most of them have many defects, such as their failure to fully use the data. Thus, it is important and challenging to study a new gas concentration prediction system given intelligent edge devices.

In this paper, a novel gas concentration prediction method is proposed to overcome the problems encountered by existing prediction

☆ This work was supported by the National Science Foundation of China (No. 61872002, No. 61873309, No. 61572137, and No. 61728202) and the Natural Science Foundation of Anhui Province of China (No. 1808085MF197), and Shanghai Innovation Action Plan Project under Grant (No. 19510710500, No. 18510760200, and No. 18510732000).

* Corresponding author at: School of Computer Science, Fudan University, Shanghai, China.
E-mail addresses: zhangyiwen@ahu.edu.cn (Y. Zhang), guohaishuai@foxmail.com (H. Guo), lzh@fudan.edu.cn (Z. Lu), zhanlu@fudan.edu.cn (L. Zhan), patrick.hung@uoit.ca (P.C.K. Hung).

https://doi.org/10.1016/j.engappai.2020.103643
Received 15 November 2019; Received in revised form 2 March 2020; Accepted 6 April 2020
Available online xxxx
0952-1976/

methods. The proposed method makes accurate gas concentration predictions based on a single hidden layer random weight neural network (SRWNN). It makes interval predictions instead of traditional point predictions because point predictions can only indicate errors but cannot reflect the probability of achieving the expectation (Xiang et al., 2016). SRWNN takes the multidimensional coal mine data as input and explores the potential connection between gas concentration and the data. To identify the optimal input data for SRWNN, we analyze the factors influencing the changes of gas concentration through Grey Correlation Analysis (GCA) (Tsai and Hsu, 2010; Wang et al., 2017). A total of 9 correlative factors are identified and selected as the candidate influential factors. Then, we prune the influential candidate factors according to Grey Correlation Degree (GCD) between the factors. Finally, the input of SRWNN is described as an eight-tuple $I ::= \langle G, C, A, T, P, D, V, M \rangle$, where G is a gas concentration time series, C is CO concentration, A is air volume, T is temperature, P is pressure in a coal mine, D is the number of continuous mining days, V is mining volume and M is mining depth. Compared with traditional backpropagation (Guo et al., 2011), a non-dominated sorting genetic algorithm II (NSGA-II) is easier to use and is more adaptable. It has achieved excellent results in training neural networks (Ak et al., 2016; Lian et al., 2016). Thus, NSGA-II (Deb et al., 2002) is employed to train SRWNN. During the training process, a certain number of randomly generated SRWNNs are regarded as a population. Then, high-performance models are obtained through repetitive crossover, mutation, sorting and selection operations.

With the rapid growth of industrial data collected by intelligent edge devices, the prediction model needs to be recalibrated frequently. Hence, it is necessary to accelerate a training process. We analyze the training algorithm and find that, when the size of the industrial coal mine data is very large, the majority of training time for SRWNN is consumed by the calculation of individual fitness. In addition, the number of intelligent edge devices is usually very large. If the prediction task is performed on multiple unrelated servers, it is difficult to achieve load balancing and to manage a large number of prediction processes. Therefore, we propose a distributed gas concentration prediction scheme (García-Valls and Lino Ferreira, 2018). Under this scheme, the prediction model is trained on Spark in a distributed manner. Stable and reliable real-time predictions are made on Storm.

In summary, our contributions in this paper are as follows:

(1) SRWNN is proposed for gas concentration prediction and is trained with the robust NSGA-II. It opens up an alternative to traditional time series prediction. Instead of point prediction made by conventional gas concentration prediction approaches, SRWNN provides the first attempt to make interval predictions.
(2) The performance of SRWNN is improved by pruning and using multidimensional industrial coal mine data to mine the potential connections between gas concentration and other data.
(3) The distributed application of the SRWNN in an intelligent edge system is realized. It reduces the time consumed for training the SRWNN in large-scale data samples, and solves the problems of large-scale real-time prediction task execution and load balancing.
(4) Extensive experiments are conducted on a large amount of real industrial data collected from the Zhujidong Coal Mine in China to evaluate the performance of SRWNN.

The remainder of this paper is organized as follows. Section 2 provides related works and problem statement. Section 3 provides preliminaries. Section 4 presents SRWNN and its training method. Section 5 introduces the distributed gas concentration prediction scheme. Section 6 presents and discusses the experimental results. Section 7 concludes this paper and points out future work.

2. Related works and problem statement

2.1. Related works

Several approaches to gas concentration prediction can be found in the current literature. Existing approaches are classified into two categories: statistical approaches (Lee and Tong, 2011; Hill et al., 2012) and simple artificial intelligence (AI) based approaches (Guo et al., 2011; Ren et al., 2014; Yao and Ge, 2016). Most of those approaches make predictions based solely on time series data and thus have many disadvantages. The time series autoregressive (AR) model (Lee and Tong, 2011) is a mature prediction method. Based on the AR model, the auto-regressive and moving average model (ARMA) (Hill et al., 2012) was developed. These prediction models are very simple and predict gas concentration with a formula for a short time series. The Back-Propagation (BP) neural network (Guo et al., 2011; Ren et al., 2014) makes gas concentration prediction based on multilayer forward feedback neural networks. However, the BP neural network requires adjustment of too many parameters and thus suffers from poor adaptability. Besides, the above methods only make point predictions, which only provide the prediction error but indicate no information about the probability of achieving the expectation. The Particle Swarm Optimization-Gravitational Search Algorithm (PSOGSA) significantly improves the performance of the traditional prediction methods (Lian et al., 2016). Through continuous iterations, the PSOGSA can efficiently find the optimal prediction model. However, this method also requires repetitive parameter adjustment.

2.2. Problem statement

Gas concentration prediction (GCP) seeks to obtain a numerical indicator of gas concentration by analyzing a set of data collected from intelligent edge devices.

Almost all the existing gas concentration prediction methods are implemented by using historical gas concentration. Suppose x_i is the gas concentration at time i, then the process of those methods implementing gas concentration prediction can be expressed as

$$x_{i+1} = g(x_i, x_{i-1}, x_{i-2}, x_{i-3} \ldots) \tag{1}$$

where g is a strategy for solving problems. The difference between different methods is only that the strategies are different.

Obviously, in these methods, no attempt was made to uncover the correlation between gas concentration and other data and apply them to gas concentration prediction. In fact, in addition to gas concentration, there are many other data in coal mines, such as carbon monoxide concentration, pressure, wind speed, temperature and so on. There is a potential relationship within such data. Through grey correlation analysis, some higher correlation data is selected. Finally, coal mine data are described as an eight-tuple I

$$I ::= \langle G, C, A, T, P, D, V, M \rangle \tag{2}$$

where G is the gas concentration time series, C is the CO concentration, A is the air volume, T is the temperature, P is the pressure in the coal mine, D is the number of continuous mining days, V is the mining volume and M is the mining depth.

In this paper, unlike formula (1), we attempt to use the eight-tuple to construct a novel gas concentration prediction model. Moreover, in order to speed up the training, and to achieve simultaneous prediction of large-scale monitoring points and maximize the role of intelligent edge devices, we present a distributed gas concentration prediction scheme.

3. Preliminaries

3.1. Single-hidden layer feed-forward neural networks

Single-hidden Layer Feed-forward Neural networks (SLFN) have achieved outstanding results recently (Liu et al., 2013; Sulistyo et al., 2017; Chen et al., 2018). A single-hidden layer feed-forward neural network consists of an input layer, a hidden layer and an output layer. Adjacent layers are fully connected. SLFNs are formulated below.

Given a data set $(\boldsymbol{x_i}, y_i)$, $i = 1, 2, \ldots, N$, where $\boldsymbol{x_i}$ is the ith sample, and y_i is the label vector. Then, the output vector $\boldsymbol{o_i}$ can be modeled as follows:

$$\boldsymbol{o}_i = \sum_{j=1}^{h} \boldsymbol{\beta}_j f(\boldsymbol{\alpha}_j \boldsymbol{x}_i + b_j)\,,\ i = 1, 2, \ldots, N \tag{3}$$

where h is the number of neurons in the hidden layer; b_j is the bias of the jth hidden layer neuron; $\boldsymbol{\alpha}_j$ is the weight vector between the jth neurons in the hidden layer and the neurons in the input layer; $\boldsymbol{\beta}_j$ is the weight vector between the jth neurons in the hidden layer and the neurons in the output layer, and f is the activation function.

3.2. Interval prediction

Most conventional prediction methods use point prediction, which predicts a point value with a time series (Wen et al., 2018; Tran et al., 2018). However, point prediction only reflects prediction error but provides no information about the probability of achieving the expectation. When it is necessary to guarantee the probability of achieving the expectation, point prediction is not suitable.

Interval prediction overcomes the limitation of point prediction by providing an interval instead of a fixed value. It is formally defined as follows:

Definition 1 (*Interval Prediction*)**.** Given a set of pairs $(\boldsymbol{x_i}, y_i)$, $i = 1, 2, \ldots, N$, where $\boldsymbol{x_i} \in R_n$, $y_i \in R$, n is the dimensionality of $\boldsymbol{x_i}$, the prediction interval (PI) $I_i = [L(\boldsymbol{x_i}), U(\boldsymbol{x_i})]$ with the nominal confidence $100(1-\alpha)\%$ can be constructed with

$$Pr\left(L\left(\boldsymbol{x_i}\right) \le y_i \le U\left(\boldsymbol{x_i}\right)\right) = 100\,(1-\alpha)\,\% \tag{4}$$

where Pr is the probability, $L(\boldsymbol{x_i})$ is the lower bound of the ith prediction interval, and $U(\boldsymbol{x_i})$ is the upper bound of the ith prediction interval, α is the probability that the expected value falls outside the prediction interval.

To evaluate prediction intervals I_i, $i = 1, 2, \ldots, N$, we introduce three widely used performance indicators, i.e., prediction interval coverage possibility (PICP), prediction interval width (PIW) and normal mean PIW(NMPIW) (Ak et al., 2016; Lian et al., 2016).

PICP represent the proportion of labels lying within the PI:

$$\mathrm{PICP} = \frac{1}{N}\sum_{i=1}^{N} c_i \tag{5}$$

where N is the size of the sample, and

$$c_i = \begin{cases} 1, & \text{if} \quad L(x_i) \le y_i \le U(x_i) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

If PI is wide, PICP can be very large. However, an overly broad PI does not provide much information. Therefore, PIW needs to be considered.

$$PIW(x_i) = U(x_i) - L(x_i) \tag{7}$$

The PIW of a single sample cannot represent the entire sample set. Thus, NMPIW is applied.

$$\mathrm{NMPIW} = \frac{1}{N(y_{\max} - y_{\min})}\sum_{i=1}^{N} \mathrm{PIW}(x_i) \tag{8}$$

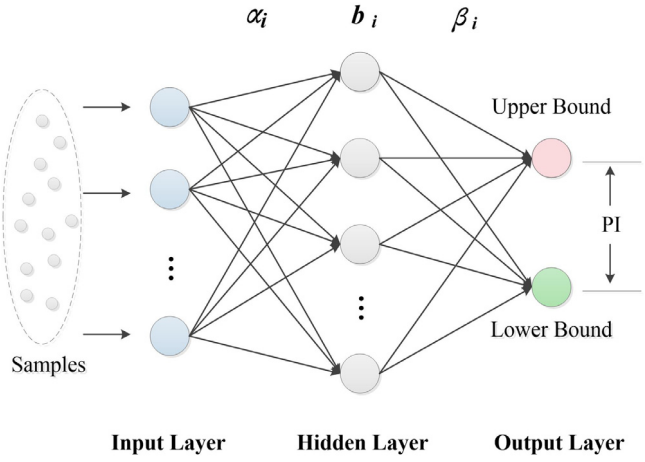

Fig. 1. Structure of SRWNN. . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where y_{max} is the maximum of y_i, $i \in [1, n]$, and y_{min} is the minimum of y_i, $i \in [1, n]$.

In practice, a higher PICP and lower NMPIW are usually preferred. However, the two objectives are mutually restrictive because NMPIW becomes larger when PICP increases. Therefore, it is important to reach a balance between a high PICP and a small NMPIW.

3.3. Multi-objective genetic algorithm: NSGA-II

The NSGA-II algorithm is a classic multi-objective genetic algorithm that is characterized by its robustness and ease of use. The goal of NSGA-II is to find the first non-dominated front (Deb et al., 2002; Yang et al., 2018a) which meets the following function:

$$\max\left(f_a(x), f_b(x), \ldots\right) \tag{9}$$

where $f_s(x)$ $(s = a, b, \ldots)$ are the objective functions to be maximized.

The solutions are compared in two ways (Deb et al., 2002): (1) dominance: if one solution dominates the other, it will be superior. If the two solutions do not have a dominant relationship, the crowding distance needs to be compared. (2) crowding distance: if there is no dominance relationship between the two solutions, their crowding distance will be measured. A greater crowding distance indicates a better solution. The dominance relationship and the crowding distance are defined below:

Definition 2 (*Dominance*)**.** Given two solutions, S_i and S_j, S_i dominate S_j, if and only if

$$\begin{aligned} &\forall a \in \{1, 2, \ldots, m\} : f_a(S_i) \geqslant f_a(S_j) \wedge \\ &\exists b \in \{1, 2, \ldots, m\} : f_b(S_i) > f_b(S_j) \end{aligned} \tag{10}$$

Definition 3 (*Crowding Distance*)**.** Given all solutions in rank n, S_a, S_b, $\ldots$, S_i, $\ldots$, the crowding distance can be expressed as

$$dis_i = \sum_{j=1}^{m} \frac{f_{ij}^{pre} - f_{ij}^{next}}{f_j^{\max} - f_j^{\min}} \tag{11}$$

where m is the number of objective functions, f_{ij}^{pre} and f_{ij}^{next} calculate the objective values of two adjacent solutions of S_i on the jth objective function. $f_j^{\min}$ and $f_j^{\max}$ represent the minimum and maximum of all the solutions in rank n on the jth objective function. It should be noted that the purpose of measuring the crowding distance is to maintain the diversity in the population.

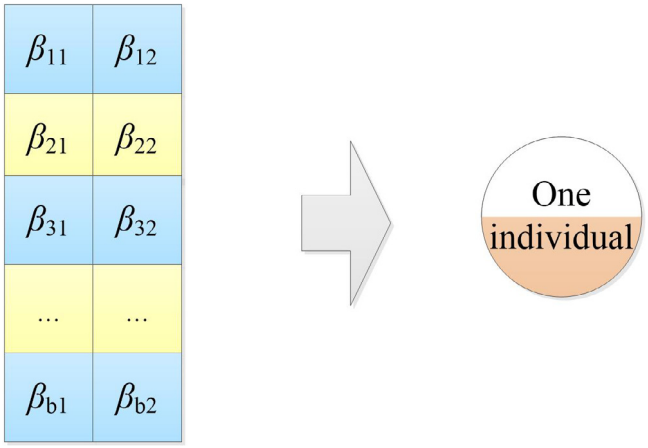

Fig. 2. Individual composition.

4. SRWNN and its training methods

4.1. The structure of SRWNN

SRWNN combines an SLFN and the lower upper bound estimation (LUBE) method (Khosravi et al., 2011). As shown in Fig. 1, the structure of an SRWNN is similar to that of an SLFN. The input of the SRWNN is a large number of samples, and the output is a prediction interval. Red neurons represent the predicted upper bound and green neurons represent the predicted lower bound. The hyperbolic tangent function is used as the activation function in the hidden layer and the logistic sigmoid function is used in the output layer. All the initial weights and biases, which are to be updated by NSGA-II, are randomly generated following the uniform distribution to leverage the advantages of NSGA-II. When the normalized data is input into the model through the input neurons, the output will be calculated with (3). Then, the SRWNN builds the prediction interval by taking the larger of the two output neurons as the upper bound of the prediction interval and the smaller neuron as the lower bound of the prediction interval. Because the safety range of gas concentration is also an interval, we can compare the prediction intervals and the safety interval to determine whether the production environment is safe.

4.2. Training SRWNN

As mentioned above, the biases of the hidden layer and the connection weights between the input layer and the hidden layer are both random values. Thus, optimizing the SRWNN becomes the problem of optimizing the connection weights between the hidden layer and the output layer. Here, we regard one SRWNN as an individual and construct a large population of individuals. As shown in Fig. 2, the connection weights between the hidden layer and the output layer constitute a weight matrix, where β_{km} is the connection weight between the kth neuron in the hidden layer and the mth neuron in the output layer. Individuals differ in their weight matrix. Thus, we regard an output weight matrix as a gene, i.e., an individual. Then, NSGA-II algorithm is employed to optimize the weight matrix. The optimization objectives of NSGA-II are the PICP and NMPIW of the SRWNN, i.e.

$$\begin{aligned} &\text{maximize(PICP) and minimize(NMPIW)} \\ &\text{Subject to: } 0 \leqslant NMPIW \leqslant 1 \\ &0 \leqslant PICP \leqslant 1 \end{aligned} \tag{12}$$

The optimal individual set can be obtained through continuous crossover, mutation, sorting and selection, as shown in Fig. 3. The size of the brown area in the circle represents the fitness of the individual. A larger brown area indicates a better individual. In each generation, half of the individuals are eliminated through sorting and selection. Between two generations, the offspring with the same population size is obtained through crossover and mutation. In the first generation, the brown area in the circle representing the fitness of an individual is usually small. As evolution progresses, the fitness of the individual increases. Eventually, we obtain high-fitness individuals in the maximum generation.

The time complexity of one generation in the training process consists of 3 major parts: (1) non-dominated sorting; (2) fitness evaluation; and (3) crossover and mutation. The time complexity in the non-dominated sorting phase is $O(MP^2)$, where M is the number of objectives and P is the population size. The time complexity in the fitness evaluation phase is $O(NP)$, where N is the number of samples. In the crossover and mutation phases, the time complexity is $O(P)$. Because M of SRWNN is 2, the total time complexity of one generation is $O(P^2 + NP)$.

5. Distributed gas concentration prediction

5.1. Architecture

Our experimental results show that the SRWNN trained by NSGA-II makes accurate gas concentration predictions, which are demonstrated in Sections 6.1 and 6.2. However, its efficiency needs to be improved, both in training and prediction. After carefully analyzing the training process of SRWNN, we find an important rule: the predominant time spent on training is the calculation of the individual's fitness while sorting, selection, crossover and mutation, which involves only the individuals and not samples, costing little time. When the number of training samples becomes large, the training time of the model is almost equivalent to the time of the fitness calculation. As more industrial data are collected, the prediction model needs to be refined frequently. Thus, it is necessary to reduce the training time. In addition, in the actual coal mine environment, the number of monitoring points that need to be predicted can be very large. Therefore, we employ a distributed cluster system to guarantee load balancing and manage a large number of prediction processes.

In recent years, several big data processing platforms have emerged, such as Hadoop, Spark, Storm, and Flink. A common and important feature of those platforms is that they allow most algorithms to be performed simultaneously on multiple machines (Zhang et al., 2018). This greatly accelerates the algorithms and reduces the computation time. We expect to implement model training (batch processing) and real-time prediction (stream computing) on a single platform to ensure the simplicity of the implementation of the SRWNN. However, there is no distributed platform that is superior at both batch processing and stream computing. Considering the characteristics of SRWNN and the advantages of different distributed platforms, we choose to implement SRWNN on Spark and Storm. Spark is a powerful distributed batch processing platform, and we trained our prediction model on Spark. Storm is an outstanding distributed platform for stream computing. Thus, it is suitable for SRWNN to make real-time gas concentration predictions on Storm. Fig. 4 shows the architecture of SRWNN implemented on Spark and Storm. The data is first pre-processed into samples, which are then cached in memory. After that, the prediction model is trained on Spark. To make predictions on Storm, the SRWNN first deploys the prediction model as bolts, and then reads real-time streaming data into spouts continuously to make predictions and obtain the prediction results in bolts. A spout is an entry for data flow into Storm and a bolt is the computing or output component of Storm.

Compared to the prediction on a single server, distributed implementation has many advantages. Firstly, a distributed training approach enables the SRWNN to learn from more samples. Secondly, Storm-based SRWNN prediction can greatly improve its real-time performance, which is very important for gas concentration predictions in coal mines. Finally, our model avoids excessive load on a single server and reduces the probability of server errors.

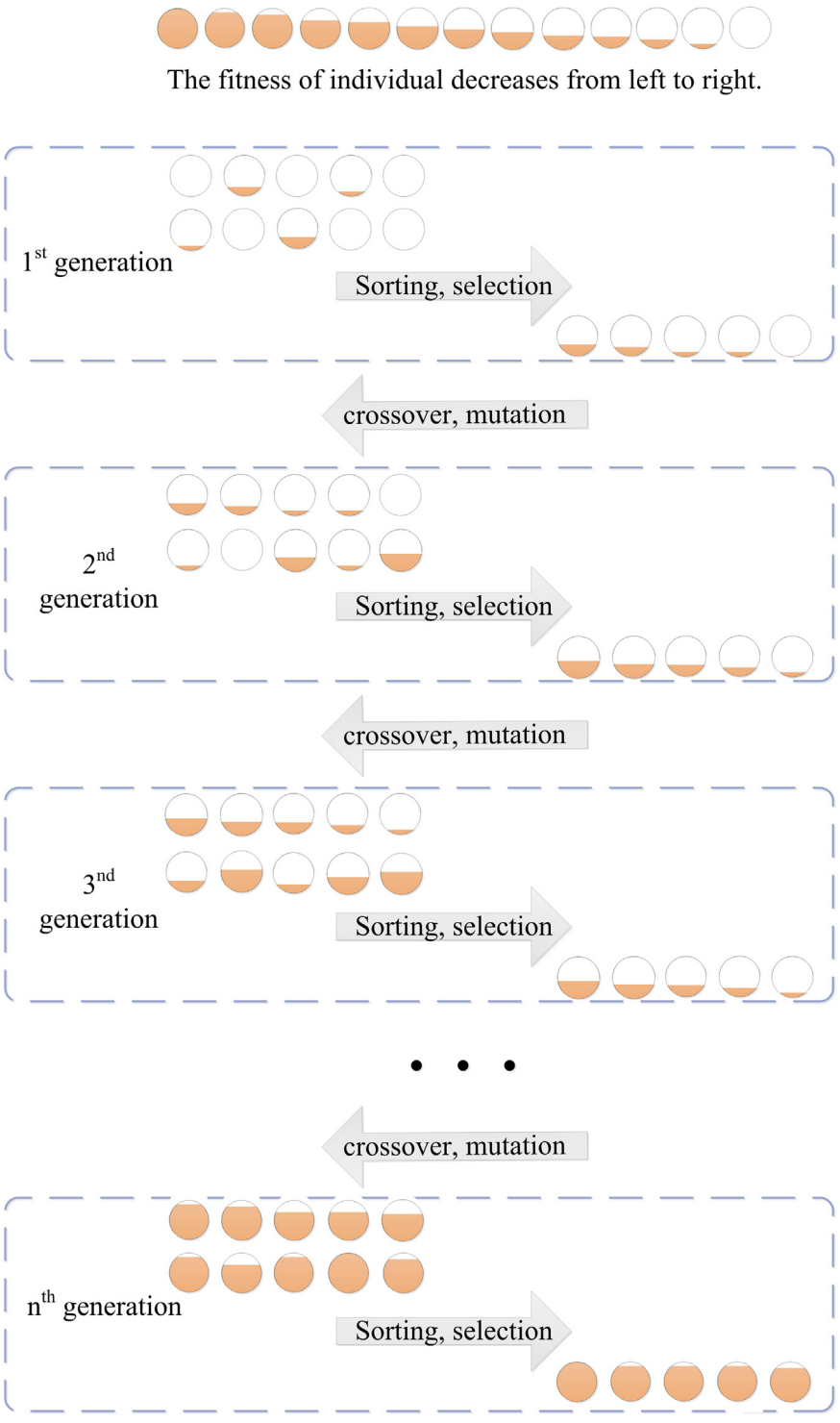

Fig. 3. The training process of SRWNN.

5.2. Distributed model training

During the training process of the SRWNN with a large number of samples, the excessive time consumption caused by the fitness calculation is the main efficiency bottleneck. Sorting, selection, crossover and mutation, which involve only individuals but do not involve samples, consume very little time. When the number of samples reaches tens of millions, the fitness calculation is extremely time-consuming. While the number of individuals in the population is generally no more than 1000, the calculation involving only the individuals and not the samples is relatively simple. To tackle this bottleneck, we train the SRWNN on Spark, which excels at performing batch processing.

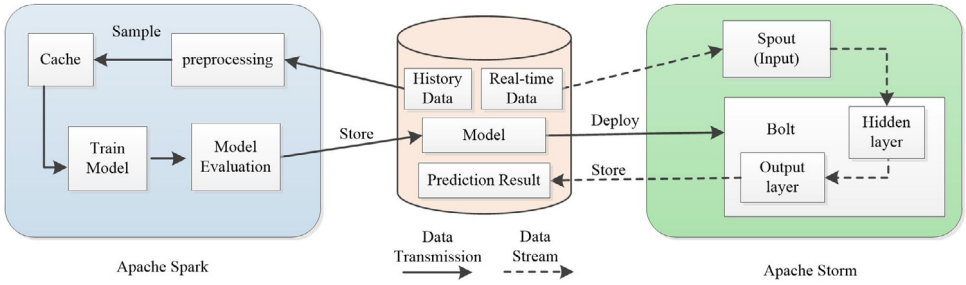

Fig. 4. Distributed gas concentration prediction scheme.

Algorithm 1: Distributed training of SRWNN

Input:

Sample set S;

Output:

Pareto-optimal SRWNN set O;

Initialization:

A random initial population P_0 ,population size N, max generation m, current generation $t = 0$, number of worker now ;

for each worker w in cluster **do**

Count sample size s_w of worker w ;

Search for the maximun label $maxLabel_w$ and minimum label $minLabel_w$ in worker w ;

end for

Total sample size $s = \sum_{i=1}^{now} s_i$;

Maximun sample label $y_{max} = \max\{ maxLabel_w \}$;

Minimun sample label $y_{min} = \min\{ minLabel_w \}$;

for $i = 1$ to N **do**

Compute PICP and NMPIW of ith individual in P_0 using s, y_{max}, y_{min} ;

end for

Sort P_0 by (8)(9)(10);

Create Q_0 by crossover and mutation;

Evolution:

while $t<m$

$R_t = P_t \cup Q_t$;

Compute PICP and NMPIW of ith individual in P_t using s, y_{max}, y_{min} ;

Sort R_t into F_1, F_2 ... with (8);

$k = 1, P_{t+1} = \emptyset$;

while($|P_{t+1}| + |F_k| \le N$)

$P_{t+1} = P_{t+1} \cup F_k$;

$k = k + 1$;

end while

Sort F_k by (8);

Fill P_{t+1} to the size of N with F_k ;

Create Q_{t+1} with P_{t+1} by crossover and mutation;

$t = t + 1$;

end while

$R_m = P_m \cup Q_m$;

Sort R_m and get F_1 ;

$O = F_1$;

As discussed in Section 3.2, the calculation of individual fitness includes the calculation of PICP and NMPIW. In a distributed environment, there are some differences in the calculation of PICP and NMPIW. As shown by (5) and (8), it is necessary to determine the total number of samples N. Existing methods running on a single server can easily calculate N by traversing the entire data set on the server. However, under the distributed environment, those methods are not applicable. To obtain N, we count the number of samples in each worker, and then sum them up with an accumulator. In addition, calculating NMPIW also requires determining the maximum and minimum values of the labels in all samples. To address this issue, we first identify the local maximum and minimum labels in each worker and then identify the global maximum and minimum labels from all the local maximums and minimums labels in each worker. Since the samples are invariant during the training process, these operations can be performed before the start of the genetic algorithm to avoid double counting the maximum and minimum. As shown in Algorithm 1, the distributed model training first calculates the size of the sample set and the label interval (i.e., the difference between the maximum label and the minimum label). Then, the fitness calculation is distributed to each node in each generation of the genetic algorithm, and the optimal solution set is finally obtained.

On the Spark platform, the workload of the fitness calculation is distributed across multiple workers to accelerate the calculation. Fig. 5 shows the details of the distributed model training. The master of Spark manages the cluster and is responsible for DAG scheduling and task scheduling. The fitness calculation in each generation is distributed across the workers, thereby reducing the total time needed to complete the fitness calculation.

In traditional prediction methods, the servers work independently and the load on each individual server is determined manually. The administrator needs to know the load on each server and decides which server should be assigned a new task. If the load is static, this approach is feasible. However, if the load is dynamic, this approach cannot properly balance it. A naive solution is to manually add or remove servers upon load changes. However, it is ineffective and inefficient in response to dynamic load. The SRWNN balances the load differently. It distributes the samples evenly and randomly to each worker to avoid data skew. During the training process, it re-partitions the training data to fix load imbalance caused by shuffle operations.

It should be noted that the distributed computing architecture shown in Fig. 5 is only applicable when the number of training samples is very large. If the number of training samples is small, the communication between the workers, as well as between the master and the workers, will cause excessive computation overheads and slows down the training process. In such cases, the distributed training method is inappropriate.

The time complexity of one generation in the distributed training process consists of 3 major parts: (1) non-dominated sorting; (2) fitness evaluation; and (3) crossover and mutation. In the non-dominated sorting phase, the time complexity is $O(MP^2)$, where M is the number of objectives and P is the population size. The time complexity in the fitness evaluation phase is $O(NP/W)$, where N is the number of samples, and W is the number of workers. In the crossover and mutation phases, the time complexity is O(P). Because the number of objectives M is 2, the total time complexity of one generation is $O(P^2 + NP/W)$.

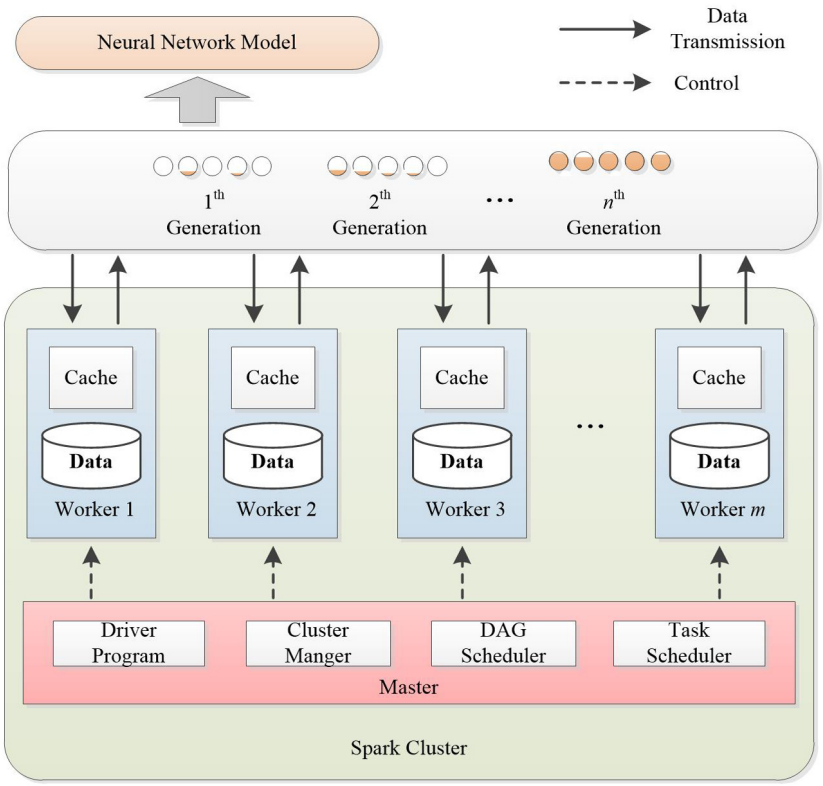

Fig. 5. Distributed training of prediction model.

5.3. Distributed real-time predictions

In actual coal mine environment, there are a large number of monitoring points, so the predicted tasks are also arduous. It is difficult for a single machine to perform all the real-time prediction tasks. However, if we use several disconnected servers to make predictions, it will not be conducive to the management of a large number of prediction processes. To address this, SRWNN makes predictions on Storm, which is a free and open distributed real-time computation platform. And Storm makes it easy to reliably process unbounded streams of data, doing for real-time processing what Spark does for batch processing. We input a large number of real-time prediction tasks to the Storm platform. As shown in Fig. 4, bolts read the trained prediction model. Then, real-time streams of industrial coal mine data are passed first into the spouts and then the bolts. The prediction tasks of SRWNN are performed in the bolts, which is divided into two parts: the calculation in the hidden layer and the calculation in the output layer. Finally, the prediction results are produced by the bolts. Since the bolts can perform calculation tasks in parallel, the calculation in the hidden layer can be performed simultaneously with the calculation of the output layer.

6. Experimental evaluation

In this section, the prediction performance of SRWNN is experimentally evaluated. The performance of centralized SRWNN is evaluated against two state-of-the-art prediction methods, i.e., PSOGSA (Lian et al., 2016) and ELM (Ak et al., 2016; Yang et al., 2018b), first on a benchmarking dataset and then on real industrial coal mine data collected from the Zhujidong Coal Mine in China. Those two sets of experiments are run on a machine with 3.60 GHz CPU and 16.00 GB RAM. Finally, we evaluated the distributed version of SRWNN on a distributed cluster system.

Table 1
Parameters for training model.

	Parameter	Value
SRWNN	Number of input neuron	5
	Number of hidden neuron	10
	Number of output neuron	2
NSGA-II	Sample size	10,000
	Population size	60
	Maximum generation	100
	Crossover probability	0.2
	Mutation probability	0.05

Table 2
Experimental comparison of benchmark.

SRWNN	Dataset 1		Dataset 2	
	PICP	NMPIW	PICP	NMPIW
1	0.916	0.492	0.908	0.507
2	0.917	0.363	0.913	0.521
3	0.903	0.442	0.895	0.483
4	0.908	0.431	0.886	0.468
5	0.896	0.491	0.916	0.533
6	0.909	0.505	0.901	0.516
7	0.917	0.486	0.920	0.541
8	0.926	0.507	0.907	0.517
9	0.938	0.476	0.914	0.509
10	0.921	0.491	0.891	0.473
Average	**0.915**	**0.468**	**0.899**	**0.491**
PSOGSA	0.901	0.497	0.888	0.589
ELM-based	0.862	0.514	0.761	0.552

6.1. Benchmark

The performance of the SRWNN is first evaluated on a benchmark with a 5-dimensional function: $y = 6x_1 + 3x_2x_3^2 - 5x_4^3 + e^{x_5}$, and two datasets are used (Deb et al., 2002). The first dataset contains 10,000

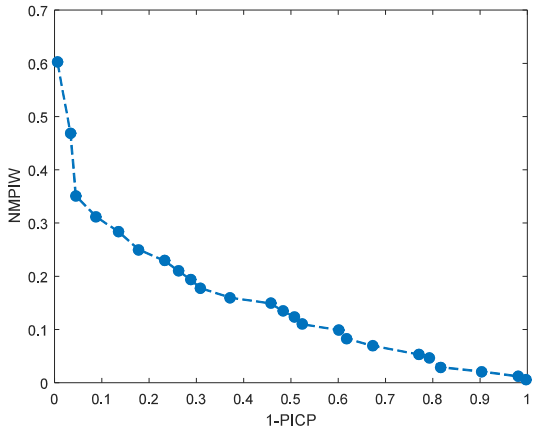

Fig. 6. NMPIW and 1-PICP of the individual in an optimal solution set.

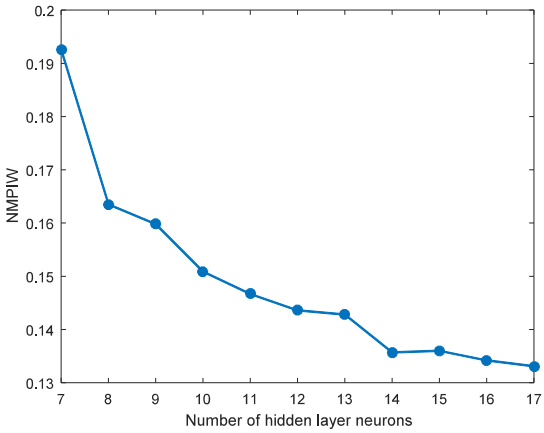

Fig. 8. Relationship between NMPIW and number of hidden layer neurons.

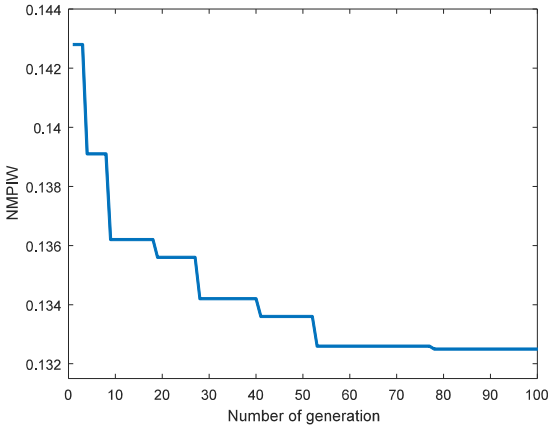

Fig. 7. Relationship between NMPIW and number of generation.

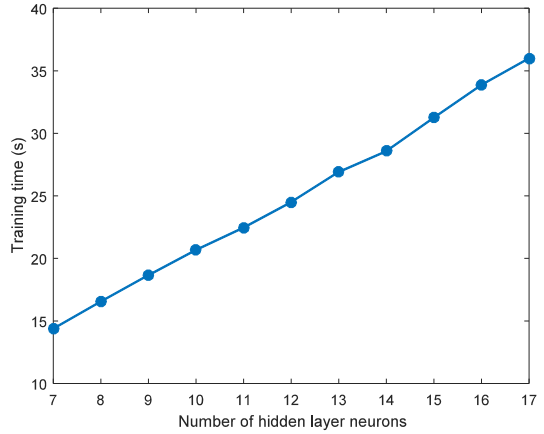

Fig. 9. Relationship between training time and the number of hidden layer neurons.

samples, all of which are samples conforming to the five-dimensional function - x_1, x_2, x_3, x_4, x_5 as inputs and y as the label. The second dataset also contains 10,000 samples. 96% of those samples conform to the 5-dimensional function and 4% is noise which adds random numbers from [−4, 4] to the original data. To evaluate the performance of the trained model, a part of the sample is selected from the dataset as testing data. Specifically, each dataset is divided into two parts: the training set (80%) and the testing set (20%).

We have run extensive experiments under settings similar to (Lian et al., 2016; Ak et al., 2016). The optimal parameters are presented in Table 1, the first three for the SRWNN and the other five for NSGA-II. Each experiment is repeated 10 times and the results are averaged.

The performance comparisons between SRWNN and the other prediction methods are reported in Table 2. When the PICP is of a certain size, SRWNN achieves lower NMPIW than other methods by 6.3% on average. Therefore, SRWNN outperforms PSOGSA and ELM-based predictions on both datasets.

6.2. Prediction with multidimensional data

In this section, we first use GCA to preprocess the data. Next, we evaluate the SRWNN on real industrial coal mine data collected from the Zhujidong Coal Mine, which is also divided into two parts: the training set (80%) and the testing set (20%). After all the data are analyzed, the data from four different coal mine locations are selected to perform the experiment, each of which includes 1.3 million samples. For convenience sake, the four datasets are denoted as Dataset A, Dataset B, Dataset C, Dataset D. The time span of the selected sample set is from April 1, 2017 to July 31, 2017. Then, we explore optimal parameters for SRWNN.

We conduct experiments and record the PICP and NMPIW of the obtained optimal solution set. The experimental results are shown in Fig. 6. The NMPIW decreases as the 1-PICP increases. When 1-PICP is less than 0.08(the PICP value is greater than 0.92), the value of NMPIW increases very rapidly. Considering the practicality of the model, in the following experiments, we set PICP = 92% to make predictions.

During the training process of the SRWNN, if the number of generations is too small, the performance of the SRWNN will be limited. With an increase in the number of generations, the training efficiency decreases. Therefore, we conduct experiments to find a suitable maximum number of generations so that the SRWNN can be efficiently trained within a reasonable training time. Fig. 7 shows the relationship between NMPIW and the number of generations. When the maximum generation is over 60, the improvement in the performance of the SRWNN is insignificant. When the maximum generation is lower than 10, the performance of the SRWNN decreases significantly. However, since the evolution process of a population is a cyclical process that repeats crossover, mutation, sorting and selection, the time consumption for evolution is proportional to the number of generations in the evolution process. To improve the training efficiency and reduce time consumption, we set the maximum generation to 60 in the following experiments.

In experiments, the number of neurons in the input layer is 15 and the number of neurons in the output layer is 2. According to the empirical formula (Huang et al., 2006): $h = \sqrt{n_i + n_o} + a$, where h is the number of hidden neurons, n_i is the number of input neurons, n_o

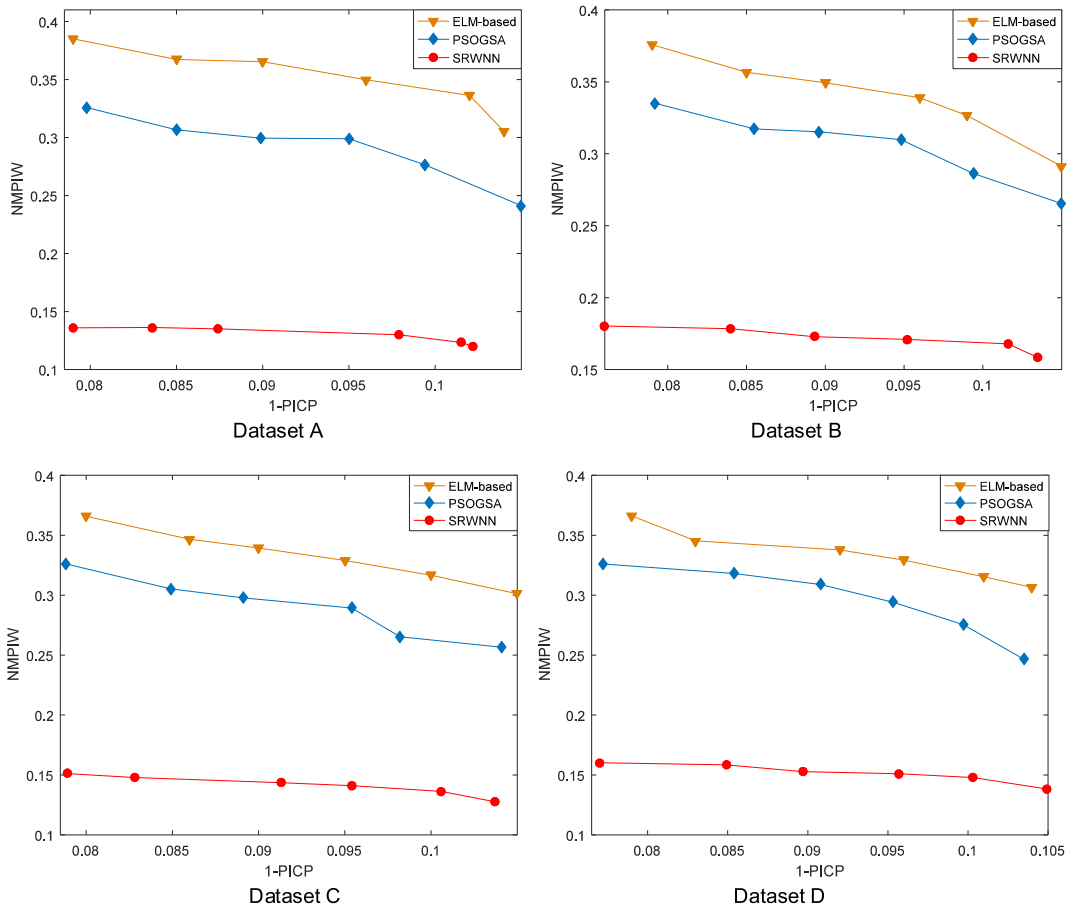

Fig. 10. Prediction performance comparison on four datasets.

is the number of output neurons, a is a constant between 1 and 10, the number of hidden neurons should be selected from the range of [5, 15]. Repeated experiments are conducted with different numbers of hidden neurons to evaluate the prediction performance of the SRWNN. And the averaged results of ten experiments are taken for the final result for each different number of hidden neurons. The results are demonstrated in Fig. 8. The number of hidden neurons changes from 7 to 17. When the number of hidden neurons is lower than 7, the neural network is oversimplified and the prediction performance is very low. Thus, the results presented in Fig. 8 start with 7 hidden neurons. When the number of hidden neurons in the SRWNN increases from 7 to 14, the achieved NMPIW value decreases significantly. However, when the number of nodes in the hidden layer is greater than 14, the decrease in NMPIW slows down and eventually converges. In addition, as shown in Fig. 9, with the increase in the number of hidden neurons, the training time of the SRWNN increases. This phenomenon is within the expected. When there are more hidden neurons in the SRWNN, the structure of the SRWNN becomes more complex, so the training time increases. Based on the conclusions drawn from Figs. 8 and 9, the number of hidden neurons is set to 14. When the number of hidden nodes is 14, the SRWNN achieves a low NMPIW value at a reasonable training time.

Next, we perform the experimental comparison between different methods. As mentioned earlier, the data we use is an 8-tuple extracted from coal mine data. An example of the 8-tuple data is shown in Table 3. Fig. 10 depicts the relationship between NMPIW and 1-PICP achieved by different prediction methods on four datasets. Across all four datasets, with a similar PICP, the SRWNN achieves a smaller NMPIW than other prediction methods by 46.3% on average. That is, to achieve the same prediction accuracy, the SRWNN needs a smaller PI. This experiment shows that the SRWNN outperforms the other two methods on real datasets.

6.3. Distributed gas concentration prediction

To evaluate the SRWNN, we compare its training time when trained on a single server and on a distributed cluster system. Then we select a distributed prediction result to reflect its prediction performance.

The training data used to train the SRWNN is real industrial data collected in the Zhujidong Coal Mine. The time span of the selected sample is from April 1, 2017 to October 31, 2017, containing 87 million samples. When we train the SRWNN on the cluster, we change the number of workers and then record the time consumption for training. All experiments are performed 10 times and the results are averaged. The experimental results are shown in Fig. 11(a). When the number of workers is between 2 and 8, the reduction in training time is significant. As the number of workers continues to increase, the rate of training time decrease starts to slow down. Fig. 11(b) illustrates the relationship between the time reduction rate and the number of workers. As shown, a larger number of workers result in a higher time reduction rate. It should be noted that when the number of workers is 1, the time reduction rate is below 0. It means that the time consumption increases rather than decreases. This is caused by the communication cost between the master and the worker incurred in the cluster.

Next, we perform a two-minute-ahead distributed gas concentration prediction. First, we train the distributed version of SRWNN. We use a piece of gas concentration data exceeding the standard level from the streaming data to reflect the ability of the SRWNN. The results are shown in Fig. 12. The blue solid line in the middle represents the actual gas concentration. The red solid line and the green solid line represent the upper bound and the lower bound, respectively, both of which are the output neurons of the SRWNN. The yellow dashed line represents the gas concentration limit. Because these data are taken from an area with low gas concentration, the limit value of

Table 3
Example of input variables.

	G (%)								C	A	T	P	D	V	M
	G1	G2	G3	G4	G5	G6	G7	G8	(‰)	(m^3/min)	(°C)	(kPa)	(day)	(t)	
1	0.87	0.87	0.87	0.86	0.85	0.87	0.87	0.92	0.124	3.5	21.6	98.4	4	329	877.3
2	0.87	0.87	0.86	0.85	0.87	0.87	0.92	0.92	0.124	3.5	21.6	98.4	4	329	877.3
3	0.87	0.86	0.85	0.87	0.87	0.92	0.92	0.92	0.131	3.5	21.7	98.4	4	329	877.3
4	0.86	0.85	0.87	0.87	0.92	0.92	0.92	0.88	0.131	3.5	21.7	98.4	4	329	877.3
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

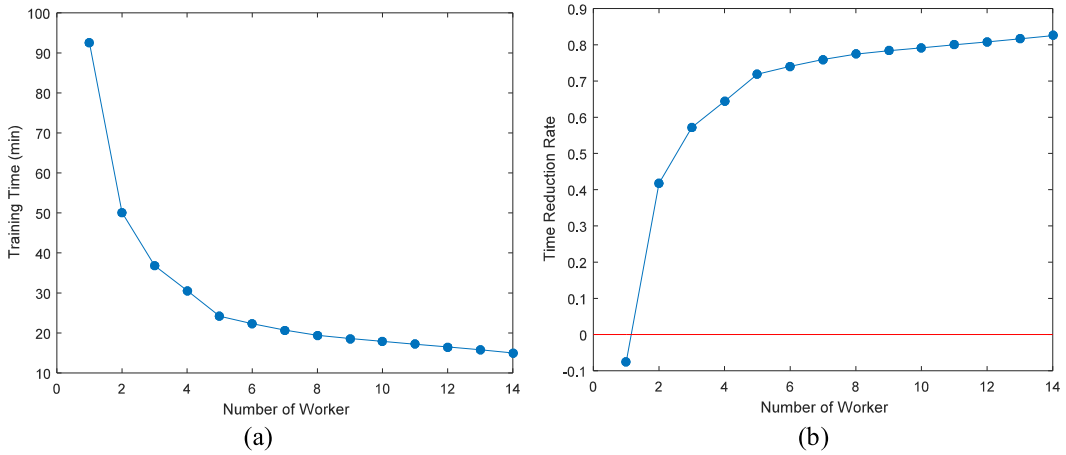

Fig. 11. Performance comparison between training on a single server and on a distributed cluster system.

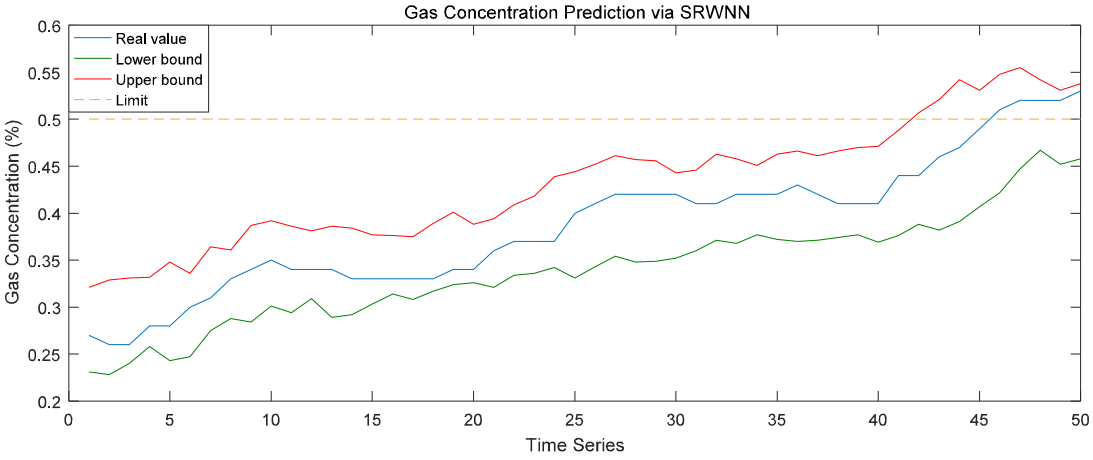

Fig. 12. Example of distributed gas concentration prediction. . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the gas concentration is 0.5%. If the gas concentration is higher than 0.5%, coal mining production must be stopped. The real value of gas concentration exceeds the limit when the time series reaches 46, and the SRWNN predicts this trend 4 time intervals in advance. The time interval between two adjacent data points is 2 min. This means that SRWNN can predict gas concentration exceeding the standard level approximately 8 min in advance. The reason for the high performance of SRWNN is that the change in gas concentration is related to not only the gas concentration time series but also many other industrial data, and SRWNN is able to take all those factors into account when making predictions.

7. Conclusion

In recent years, more and more coal mines introduce intelligent edge devices to build an intelligent edge system. It is challenging to develop a more efficient and accurate gas concentration prediction system in such intelligent edge system environment. In this paper, we propose a novel gas concentration prediction model named SRWNN for an intelligent edge system. We employ NSGA-II as a multi-objective genetic algorithm to train the prediction model, which significantly simplifies the training process. Compared with the traditional gas concentration prediction methods based on time series analysis, SRWNN considers the impact of multidimensional coal mine data on gas concentration, making the prediction results more reliable. To reduce the considerable training time consumption and make real-time predictions in large coal mines, we propose a distributed gas concentration prediction scheme. Under the scheme, the prediction model is trained on Spark and predictions are made on Storm, both in a distributed manner. Finally, extensive experiments are performed on real industrial data collected from a Chinese company. The results demonstrate the superior performance of SRWNN over state-of-the-art prediction methods. Some recent approaches such as dendritic neuron models (Tran et al., 2018) and advanced training approaches such as Savaglio et al. (2017), Sun et al.

(2018), Kang et al. (2018), Gao et al. (2018, 2019), Li et al. (2019), Savaglio et al. (2019) and Yu et al. (2019) should be studied as our future work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yiwen Zhang: Resources, Writing - review & editing, Supervision. **Haishuai Guo:** Conceptualization, Methodology, Investigation, Writing - original draft. **Zhihui Lu:** Data curation, Writing - review & editing, Supervision. **Lu Zhan:** Visualization, Writing - review & editing. **Patrick C.K. Hung:** Software, Writing - review & editing.

References

Ak, R., Fink, O., Zio, E., 2016. Two machine learning approaches for short-term wind speed time-series prediction. IEEE Trans. Neural Netw. Learn. Syst. 27, 1734–1747.

Chao, N., Longqing, S., Lele, X., Peihe, Z., Shulei, W., Yanhong, Z., 2015. Study on accidents classification of coal mine from 2001 to 2013. Saf. Coal Mines 46, 208–211.

Chen, Y., Song, S., Li, S., Yang, L., Wu, C., 2018. Domain space transfer extreme learning machine for domain adaptation. IEEE Trans. Cybern. 49, 1909–1922.

Chen, X., Tang, S., Lu, Z., Wu, J., Duan, Y., Huang, S.-C., Tang, Q., 2019. IDiSC: A new approach to IoT-data-intensive service components deployment in edge-cloud-hybrid system. IEEE Access 7, 59172–59184.

Cheng, Y., Wang, L., Liu, H., Kong, S., Yang, Q., Zhu, J., Tu, Q., 2015. Definition, theory, methods, and applications of the safe and efficient simultaneous extraction of coal and gas. Int. J. Coal Sci. Technol. 2, 52–65.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182–197.

Duan, Y., Lu, Z., Zhou, Z., Sun, X., Wu, J., 2019. Data privacy protection for edge computing of smart city in a DIKW architecture. Eng. Appl. Artif. Intell. 81, 323–335.

Gao, K., Cao, Z., Zhang, L., Chen, Z., Han, Y., Pan, Q., 2019. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. IEEE/CAA J. Autom. Sin. 6, 904–916.

Gao, S., Zhou, M., Wang, Y., Cheng, J., Yachi, H., Wang, J., 2018. Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction. IEEE Trans. Neural Netw. Learn. Syst. 30, 601–614.

García-Valls, M., Lino Ferreira, L., 2018. Introduction to the special section on real time computing and distributed systems. J. Syst. Archit. 83, 32–33.

Guo, Z.-h., Wu, J., Lu, H.-y., Wang, J.-z., 2011. A case study on a hybrid wind speed forecasting method using BP neural network. Knowl.-Based Syst. 24, 1048–1056.

Hill, D.C., McMillan, D., Bell, K.R., Infield, D., 2012. Application of auto-regressive models to UK wind speed data for power system impact studies. IEEE Trans. Sustain. Energy 3, 134–141.

Huang, G.-B., Zhu, Q.-Y., Siew, C.-K., 2006. Extreme learning machine: Theory and applications. Neurocomputing 70, 489–501.

Janjua, Z.H., Vecchio, M., Antonini, M., Antonelli, F., 2019. IRESE: An intelligent rare-event detection system using unsupervised learning on the IoT edge. Eng. Appl. Artif. Intell. 84, 41–50.

Kang, Q., Song, X., Zhou, M., Li, L., 2018. A collaborative resource allocation strategy for decomposition-based multiobjective evolutionary algorithms. IEEE Trans. Syst. Man Cybern.: Syst. 49, 2416–2423.

Khosravi, A., Nahavandi, S., Creighton, D., Atiya, A.F., 2011. Lower upper bound estimation method for construction of neural network-based prediction intervals. IEEE Trans. Neural Netw. 22, 337–346.

Lee, Y.-S., Tong, L.-I., 2011. Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. Knowl.-Based Syst. 24, 66–72.

Li, J., Pan, Q., Duan, P., Sang, H., Gao, K., 2019. Solving multi-area environmental/economic dispatch by pareto-based chemical-reaction optimization algorithm. IEEE/CAA J. Autom. Sin. 6, 1240–1250.

Lian, C., Zeng, Z., Yao, W., Tang, H., Chen, C.L.P., 2016. Landslide displacement prediction with uncertainty based on neural networks with random hidden weights. IEEE Trans. Neural Netw. Learn. Syst. 27, 2683–2695.

Liu, C.-Y., Chen, C., Chang, C.-T., Shih, L.-M., 2013. Single-hidden-layer feed-forward quantum neural network based on Grover learning. Neural Netw. 45, 144–150.

Lu, Z., Wang, N., Wu, J., Qiu, M., 2018. IoTDeM: An IoT Big Data-oriented MapReduce performance prediction extended model in multiple edge clouds. J. Parallel Distrib. Comput. 118, 316–327.

Ren, C., An, N., Wang, J., Li, L., Hu, B., Shang, D., 2014. Optimal parameters selection for BP neural network based on particle swarm optimization: A case study of wind speed forecasting. Knowl.-Based Syst. 56, 226–239.

Satyanarayanan, M., 2017. The emergence of edge computing. Computer 50, 30–39.

Savaglio, C., Fortino, G., Ganzha, M., Paprzycki, M., Bădică, C., Ivanović, M., 2017. Agent-based computing in the internet of things: a survey. In: International Symposium on Intelligent and Distributed Computing. Springer, pp. 307–320.

Savaglio, C., Gerace, P., Di Fatta, G., Fortino, G., 2019. Data mining at the iot edge. In: 2019 28th International Conference on Computer Communication and Networks (ICCCN). IEEE, pp. 1–6.

Sulistyo, S.B., Wu, D., Woo, W.L., Dlay, S.S., Gao, B., 2017. Computational deep intelligence vision sensing for nutrient content estimation in agricultural automation. IEEE Trans. Autom. Sci. Eng. 15, 1243–1257.

Sun, J., Gao, S., Dai, H., Cheng, J., Zhou, M., Wang, J., 2018. Bi-objective elite differential evolution algorithm for multivalued logic networks. IEEE Trans. Cybern. 50, 233–246.

Tran, D.T., Iosifidis, A., Kanniainen, J., Gabbouj, M., 2018. Temporal attention-augmented bilinear network for financial time-series data analysis. IEEE Trans. Neural Netw. Learn. Syst. 30, 1407–1418.

Tsai, M.-S., Hsu, F.-Y., 2010. Application of grey correlation analysis in evolutionary programming for distribution system feeder reconfiguration. IEEE Trans. Power Syst. 25, 1126–1133.

Wang, K., Xu, C., Zhang, Y., Guo, S., Zomaya, A.Y., 2017. Robust big data analytics for electricity price forecasting in the smart grid. IEEE Trans. Big Data 5, 34–45.

Wen, Y., Wu, J., Zhou, Q., Tseng, T.-L., 2018. Multiple-change-point modeling and exact Bayesian inference of degradation signal for prognostic improvement. IEEE Trans. Autom. Sci. Eng. 16, 613–628.

Wu, Z., Lu, Z., Hung, P.C., Huang, S.-C., Tong, Y., Wang, Z., 2019. QaMeC: A QoS-driven IoVs application optimizing deployment scheme in multimedia edge clouds. Future Gener. Comput. Syst 92, 17–28.

Xiang, Y., Liu, J., Liu, Y., 2016. Robust energy management of microgrid with uncertain renewable generation and load. IEEE Trans. Smart Grid 7, 1034–1043.

Xiao, A., Lu, Z., Li, J., Wu, J., Hung, P.C., 2019. SARA: Stably and quickly find optimal cloud configurations for heterogeneous big data workloads. Appl. Soft Comput. 85, 105759.

Xiong, W., Lu, Z., Li, B., Wu, Z., Hang, B., Wu, J., Xuan, X., 2019. A self-adaptive approach to service deployment under mobile edge computing for autonomous driving. Eng. Appl. Artif. Intell. 81, 397–407.

Yang, H.-F., Dillon, T.S., Chang, E., Chen, Y.-P.P., 2018b. Optimized configuration of exponential smoothing and extreme learning machine for traffic flow forecasting. IEEE Trans. Ind. Inf. 15, 23–34.

Yang, C., Ding, J., Jin, Y., Wang, C., Chai, T., 2018a. Multitasking multiobjective evolutionary operational indices optimization of beneficiation processes. IEEE Trans. Autom. Sci. Eng. 16, 1046–1057.

Yao, L., Ge, Z., 2016. Locally weighted prediction methods for latent factor analysis with supervised and semisupervised process data. IEEE Trans. Autom. Sci. Eng. 14, 126–138.

Yu, Y., Gao, S., Wang, Y., Todo, Y., 2019. Global optimum-based search differential evolution. IEEE/CAA J. Autom. Sin. 6, 379–394.

Zhang, Y., Zhou, Y., Wang, F., Sun, Z., He, Q., 2018. Service recommendation based on quotient space granularity analysis and covering algorithm on Spark. Knowl.-Based Syst. 147, 25–35.