J Sign Process Syst
DOI 10.1007/s11265-016-1119-4

An Integrated Data Preprocessing Framework Based on Apache Spark for Fault Diagnosis of Power Grid Equipment

Weiwei Shi[1] · Yongxin Zhu[1] · Tian Huang[1] · Gehao Sheng[1] · Yong Lian[1] · Guoxing Wang[1] · Yufeng Chen[2]

Received: 13 October 2015 / Revised: 28 December 2015 / Accepted: 17 February 2016

Abstract Big data techniques have been applied to power grid for the prediction and evaluation of grid conditions. However, the raw data quality can rarely meet the requirement of precise data analytics since raw data set usually contains samples with missing data to which the common data mining models are sensitive. Besides, the raw training data from a single monitoring system, e.g. dissolved gas analysis (DGA), are rarely sufficient for training in the form of valid instances since raw data set usually contains samples with noisy data. Though classic methods like neural network can be used to fill the gaps of missing data and classify the fault type, their models often fail to fit the rules of power grid conditions. This paper presents an integrated data preprocessing framework (DPF) based on Apache Spark to improve the prediction accuracy for data sets with missing data points and classification accuracy with noise data as well as to meet the big data requirement, which mainly combines missing data prediction, data fusion, data cleansing and fault type classification. First, the prediction model is trained based on the linear regression (LinR). Afterwards, we propose an optimized linear method (OLR) to improve the prediction accuracy. Then, to better utilize the strong correlation among different data sources, new data features extracted by persons correlation coefficient (PCC) are fused into a training data set. Next, principal component analysis (PCA) is taken to reduce the side effect brought by the new feature as well as retaining significant information for classification. Finally, the classification model based on logistic regression (LogR) and support vector machine (SVM) is trained to classify the fault type of electric equipment. We test the DPF framework on missing data prediction and fault type classification of power transformers in power grid system. The experimental results show that the predictors based on the proposed framework achieve lower mean square error and the classifiers obtain higher accuracy than traditional ones. Besides, the training time required for training large-scale data shows a decreasing trend. Therefore, the data preprocessing framework DPF would be a good candidate to predict the missing data and classify the fault type in power grid system.

✉ Yongxin Zhu
zhuyongxin@sjtu.edu.cn

Weiwei Shi
iamweiweishi@sjtu.edu.cn

Tian Huang
ian_malcolm@sjtu.edu.cn

Gehao Sheng
shenghe@sjtu.edu.cn

Yong Lian
eleliany@sjtu.edu.cn

Guoxing Wang
guoxing@sjtu.edu.cn

Yufeng Chen
chenyufeng169@sina.com

[1] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China

[2] Electric Power Research Institute of Shandong Power Supply Company of State Grid, Shandong, China

Keywords Big data · Apache spark · Framework · Missing data prediction · Fault diagnose

1 Introduction

The quality of data preprocessing is regarded as a significant issue of industrial process, market success and decision-making activities after initial successes of big data techniques in internet business [1–3]. The main reason is that the quality of the data inputted into the training model may strongly influence the results of the data analysis. Generally speaking, appropriate data preprocessing makes data analysis easy, while inappropriate data preprocessing is difficult for reasonable analyzing. On the other hand, the amount of data produced by modern data acquisition techniques is so big that it makes data preprocessing a time-consuming task [4]. It has been claimed that more than 50 percent of the time and effort in data analysis projects is required for data preprocessing [5]. Therefore, keeping the total preprocessing time in a reasonable range becomes researchers immediate concern.

Data preprocessing usually refers to processing the input data that contains the missing data, noise data and redundant data. Various interpretations have been given to the role and the need for data preprocessing [6–8]. First, a large collection of data mining and statistical methods have been proposed to improve data quality due to missing data. For example, Ma's team proposed a good method for missing data prediction [9]. The algorithm focused on recommender systems using improved collaborative filtering method which outperformed the traditional collaborative filtering method. Nogueira et al. [10] solved a practical problem based on the Fast Fuzzy Clustering Algorithm in the real world: the prediction of bankruptcy, in which the used data set has missing values. Lei and Wang presented a method for preprocessing the missing observed data by adopting the multiple imputation technique for Macau air pollution index (API) prediction using the Adaptive Neuro-Fuzzy Inference System (ANFIS) [11]. The API forecasting performance after missing data pre-processing is better than the conventional case without preprocessing. Also, a great number of methods have been utilized to suppress the noise in the data. Tian proposed a wavelet threshold de-noising method of data preprocessing and presented a new threshold function, which inherited the advantages of both two improved threshold functions, and introduced the reconstruction factor to select wavelet basis function [12]. The new threshold function has better de-noising effect through the simulation, and step feature is more obvious, convenient for subsequent data analysis. Wei et al. described a noise data processing approach based on rigid structure matching to better capture human motion [13]. Experiments show that the algorithm presented can effectively implement motion data processing. Silva et al. proposed a preprocessing methodology that combined the Principal Component Analysis (PCA) method with additive Gaussian noise as a way to prevent conflicts between correlated variables suffering perturbations of decorrelated noise and reduce the dimensions at the same time [14]. In all experiments carried out the achieved results are significantly better than the performance achieved without preprocessing. All these strong evidence reveals that data preprocessing has a significant influence on training models and the subsequent data analysis.

Actually, in power grid systems, data missing happens so frequently due to the harsh working condition of sensors that classic methods often fail to handle. Expensive critical equipment such as main power transformers is monitored by multiple sensors. Unfortunately, these sensors are not as reliable as the equipment in the harsh open air working condition under the workload of 7×24 hours. Moreover, sensors in remote rural areas such as mountains are usually maintained at an even worse level by workers who received less training than workers in city. Thus, it is normal and inevitable for the sensor system to produce flaw data sets, which lost or hidden some necessary information. On the other hand, the large-scale multi-dimensional data gathered from different data collection facilities also become a challenge to the traditional data analysis methods or tools [15]. These losses affect the data quality so badly that classic data mining and statistical methods alone cannot process these data properly. Therefore, the de-nosing of the original data and such data quality improvement methods turn out to be necessary for effective data analysis in power grid system.

In the past decade, the amount of data that needs to be processed has been increasing exponentially since we are constantly being told that we live in the Information Era - the Age of Big Data [16]. Although traditional methods show barely satisfactory performance, they cannot effectively deal with big data as a result of improvements in data generation and storage capacity [17]. First, analyzing big data is a complex and time-consuming task, which needs more efficient and specific analysis tool than traditional ones. Second, effective data preprocessing can increase generalization ability of data analysis, especially for the big data. Third, data preprocessing can largely reduce model complexity of training models, which are important for big data analysis tasks. Essentially, the main purpose of data analysis is to discover valuable knowledge that will be used to suggest conclusions, make decisions and solve such problems [18], but the poor quality of the inputted data may prevent this. In most cases, defects within the original big data are not noticed until the data analysis starts. Therefore, data preprocessing in big data becomes a very important issue, which is also highly recognized in the power grid system.

From the overview above, we found that there are three main problems in the data analysis of power grid systems. The first is that there is no universal methodology or framework for big data analysis in power grid systems to date. That is to say, there is no systematic and suitable data preprocessing architecture for big data analysis. Although some related researches are presented at present, a systematic and methodical work about big data preprocessing has not been formulated so far. Most existing studies focus only on one aspect of data preprocessing. Therefore, it is necessary to construct a pervasive data preprocessing framework for big data analysis, which could be used in power grid system. Second, with the data of power grid system sharply massing today, the conventional data mining statistical methods usually cannot effectively process the input data due to the massive volumes of data. That is to say, although the rich and abundant data resources provide more information for better analysis, they also bring more difficulties in the data preprocessing and subsequent analysis. Thus, it is essential to optimize the traditional methods and modify them more adaptable in power grid system. Third, in the data preprocessing for big data analysis, some important dilemmas and issues often appeared and are hard to handle. That is, in big data backdrop, the preprocessing is also a time-consuming task. And it is difficult to decide whether the efforts spent on preprocessing would lead to better performance and whether or not the spent time and cost is worthwhile.

In light of the three problems outlined above, the main contributions of this paper are as follows:

1. We propose an integrated data preprocess framework base on Apache Spark for big data analysis in power grid system;
2. We propose an optimized missing data prediction algorithm for better accuracy;
3. We propose an alternative prediction strategy utilizing different features for improving prediction accuracy;
4. We give an alternative solution for multiclass classification base on the binary classification in Apache Spark;
5. We analyze and confirm the effects of the proposed data preprocessing framework on missing data prediction and fault diagnosis.

Unlike previous work [19, 20], our proposed data preprocessing framework mainlyfocuses on dealing with the problem of big data analysis. Thus, we propose a different optimized linear regression algorithm for missing data prediction, which is proved to be more suitable than previous ones. Also, the existing algorithm in Apache Spark platform, like support vector machine, cannot be directly applied to our case. Thus it is modified to adapt to our specific situation. Besides, we also integrate the missing data prediction and fault classification into one framework, which is of great practical significance.

The remainder of the study is organized as follows: A brief description of methodology preliminary for data preprocessing is presented in Section 2. In Section 3, the steps of the fault diagnose framework in every phase, as well as important issues of the integrated scheme, are described and discussed in detail. Accordingly, some intelligent optimizations to some important issues and dilemmas are provided. A comprehensive analysis of the framework for predicting the missing data originated in power grid systems and diagnosing the fault type of electric equipment is proposed in Section 4. Finally, the paper ends with concluding remarks and future directions in Section 5.

2 Methodology Preliminary

In terms of data preprocessing, there are already a number of researches evidenced by [6, 17, 21]. As for predicting missing data, linear regression is an alternative method and could be conveniently realized on big data platforms. But it needs to be optimized for better performance. As for fault type classification, logistic regression and support vector machine are both alternative. However, the existing SVM classification algorithm on Apache Spark can only deal with binary classification problems till now. Thus it needs to be further modified to adapt to our case. In this section, we will present relevant methods in detail and give an introduction to Apache Spark platform.

2.1 Methods for Missing Data Prediction

2.1.1 An Illustration Example of Missing Data Prediction

Taking the situation shown in Fig. 1 into consideration, we assume that y_t is missing. The time series model for missing data prediction is as follows [19] :

$$y_t = f(y_{t-1}, y_{t-2}, y_{t-3}) + \varepsilon_t \quad (1)$$

where $f()$ is a nonlinear function and ε_t stands for additive noise. The goal of the problem is to predict y_t based on past content, that is y_{t-1} y_{t-2}, and y_{t-3}.

2.1.2 Linear Regression

Simple ways such as eyeballing, calculating appropriate mean value and interpolation could be used in missing data prediction [21]. Although these methods are useful in some particular situations, they might fail when it comes to complex situations, such as large scale missing data prediction [22].

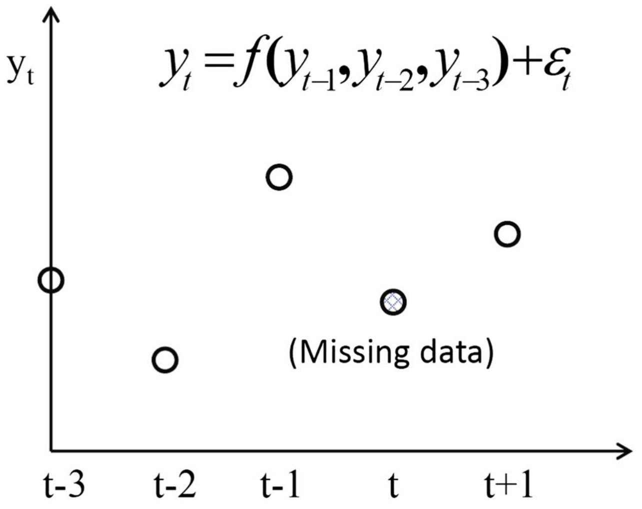


Figure 1 Illustration of missing data prediction.

In statistics, linear regression (LinR) is an approach for modeling the relationship between a scalar dependent variable and one or more explanatory variables [23]. Given a set of training data $\{X, Y\} = \{(X_1, y_1), \dots, (X_N, y_N)\}$ from which to estimate the $\hat{y}$ given a new input vector X_{in}. Each $X_i = (x_{i1}, x_{i2}, \dots, x_{im})^T$ is a vector of feature measurements for the ith entity. The linear regression model has the form

$$y_i = \beta_1 x_{i1} + \dots + \beta_m x_{im} + \varepsilon_i, i = 1, \dots, N, \tag{2}$$

where the β_t's, t= 1, 2, . . ., m, are unknown parameters, and the ε_t is a disturbance term. In matrix notation, the predicted value when given X_{in} is

$$\hat{y} = X_{in}(X^T X)^{-1} X^T y, \tag{3}$$

where the term $(X^T X)^{-1} X^T$ is known as pseudo-inverse of X. The detailed solution for LinR could be found in [23].

2.1.3 Optimized Linear Regression

Assume that we have predicted value $\hat{y}$ with respect to the input X_{in}. Apparently, the prediction square error is

$$error = |y - \hat{y}|^2. \tag{4}$$

Our optimization objective is to diminish the error further based on the fundamental linear regression model. Our proposed strategy is as follows:

$$y_i' = \begin{cases} y_i' + \sigma & if \quad error_i < -\lambda \\ y_i' & if \quad -\lambda \le error_i \le \lambda \\ y_i' - \sigma & if \quad error_i > -\lambda \end{cases} (i = 1, \dots, n). \tag{5}$$

First, given a positive threshold λ, if the predicted error of ith sample is less than $-\lambda$, we add one positive penalty factor σ to original value y_i. On the contrary, if the predicted error is larger than λ, original value y plus is the new y_i'. Otherwise, we maintain y_i constant. Then, a new data set $D' = (y', X)$ is constructed.

According to least mean squares (LMS) updating rule, which is known as Widrow-Hoff learning rule, the parameter β is obtained by

$$\beta_j' := \beta_j + C \sum_{i=1}^{m} (y^{(i)} - h_\beta(x^{(i)}))x_j^{(i)} \tag{6}$$

where C is the learning rate and the h is the represent functions. We suppose that we are dealing with the situation when the prediction error is less than $-\lambda$ and then this updating rule is represented as

$$\begin{aligned} \beta_j' : &= \beta_j + C \sum_{i=1}^{m} (y'^{(i)} - h_\beta(x^{(i)}))x_j^{(i)} \\ &= \beta_j + C \sum_{i=1}^{m} (y^{(i)} + \sigma - h_\beta(x^{(i)}))x_j^{(i)} \\ &= \beta_j + C \sum_{i=1}^{m} (y^{(i)} - h_\beta(x^{(i)}))x_j^{(i)} + \sigma x_j^{(i)} \end{aligned} \tag{7}$$

Next, the prediction result is

$$\begin{aligned} & h_{\beta'}(x^{(i)}) \\ &= \sum_{j=0}^{n} \beta_j' x_j^{(i)} \\ &= \sum_{j=0}^{n} (\beta_j + C \sum_{i=1}^{m} (y^{(i)} - h_\beta(x^{(i)}))x_j^{(i)} + \sigma x_j^{(i)})x_j^{(i)} \\ &= \sum_{j=0}^{n} (\beta_j + C \sum_{i=1}^{m} (y^{(i)} - h_\beta(x^{(i)}))x_j^{(i)})x_j^{(i)} + \sigma x_j^{(i)^2} \\ &= h_\beta(x^{(i)}) + C\sigma \sum_{j=0}^{n} x_j^{(i)^2}. \end{aligned} \tag{8}$$

Next, based on the new data set D', we can obtain a new linear regression model and new parameters β'. Finally, the prediction difference of the ith input between the new model and the original model is

$$\begin{aligned} diff(x^{(i)}) &= h_{\beta'}(x^{(i)}) - h_\beta(x^{(i)}) \\ &= h_\beta(x^{(i)}) + C\sigma \sum_{j=0}^{n} x_j^{(i)^2} - h_\beta(x^{(i)}) \\ &= C\sigma \sum_{j=0}^{n} x_j^{(i)^2}. \end{aligned} \tag{9}$$

Thus, considering that the premise is that the original prediction error is less than $-\lambda$ and the learning rate C and penalty factor σ are both positive, it is reasonable to deduce that if the threshold λ and the penalty σ are carefully and appropriately chosen, the newly obtained absolute value of prediction error can be substantially less than the original prediction error. Similarly, when it comes to the situation when the prediction error is larger than λ, the prediction error can also be reduced based on the strategy. In general, the optimized model can outperform the original one.

2.2 Methods for Fault Type Classification

2.2.1 Logistic Regression for Classification

Logistic Regression (LogR) is used to model the posterior probabilities of the K classes via linear functions in x [24], while ensuring that the probabilities sum to one and remain in [0, 1] at the same time. The model form is as follows:

$$\log \frac{\Pr(Y=t|X=x)}{\Pr(Y=K|X=x)} = \varepsilon_t + \beta_t^T x, t = 1, 2, ..., K-1, \tag{10}$$

where the β_t^T are unknown coefficients, and the ε_t is a disturbance term. The regression coefficients are usually estimated using maximum likelihood estimation. The likelihood for N samples is

$$l(\theta) = \prod_{i=1}^{N} p_{yi}(x_i; \theta), \tag{11}$$

where $p_{yi}(x_i;\theta) = Pr(Y = y_i|X = x_i;\theta)$. The optimization problem (11) does not have an analytic solution, but it has a global optimal solution due to the fact that the log-likelihood $log(l(\theta))$ is a convex function. Further details on the LogR can be found in [24].

2.2.2 A Short Review of SVM for Multi-Class Classification

Vapnik and his coworkers originally developed Support Vector Machine (SVM) in the 1990s [25], which is proved to be successful in many significant fields. SVM has the potential to handle large feature spaces, because the training of SVM is carried out so that the dimension of classified vectors does not has as distinct influence on the performance of SVM as it has on the performance of conventional classifier. Here we briefly introduce SVM for multi-class classification, which can be divided into three categories [26].

1. One-versus-all (OVA): Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class. The earliest method OVA for the K-class classification problem is to train K independent binary classifiers that are each trained to distinguish training samples for one class with regard to remaining classes. The OVA method is very simple to implement, relatively fast running, and produces results that are often as accurate as other methods.
2. One-versus-one (OVO): Another method is called One-versus-one (OVO) [28]. To solve the K-class classification problem, the OVO method constructs $K(K-1)/2$ classifiers where each one is trained on data from two classes. For training data from the ith and the jth classes, we solve the binary classification problem. This method has the same simple conceptual justification as OVA.
3. Binary tree support vector machine (BTSVM): BTSVM decomposes multi-class problems into a series of same two-type classification problems, which separates a pair of macro-classes with conventional SVM classifier [26]. The traditional structure is shown in Fig. 2 with K equal to 4. The steps of BTSVM are as follows. First, we label the original data into different groups according to the raw labels. For example, we set Class D1 as a set of positive samples and set the rest samples as the negative sample. Then, an SVM C1 is constructed. Likewise, we use the same way for classifying all the multi-class training data. When we predict new input data sets, we only need to input the SVMs layer by layer till we get a classification result, that is to say, till we get to the leaf node on the binary tree. BTSVM has a relatively simple structure and only needs to construct K-1 SVM classifiers to the case of K classes, which made it to be a potential candidate for solving multi-class classification problems. Thus, we adopt the BTSVM approach for building multi-class classification models in our framework.

2.3 Apache Spark

Spark is a distributed computing framework developed at UC Berkeley AMPLab. Spark's in-memory parallel execution model in which all data will be loaded into memory avoids the I/O bottleneck and thus benefits the iterative computation [27]. Spark also provides very flexible DAG-based (directed acyclic graph) data flows, which can significantly speedup the computation of the iterative algorithms. The two features of Spark bring performance up to 100 times faster compared to Hadoops two-stage MapReduce paradigm [28].

The core of the programming model is called resilient distributed data sets (RDD), a new distributed memory abstraction [29]. The RDD is an immutable collection of data records, combined with a partitioning function, which assigns elements of the collection to partitions. There are

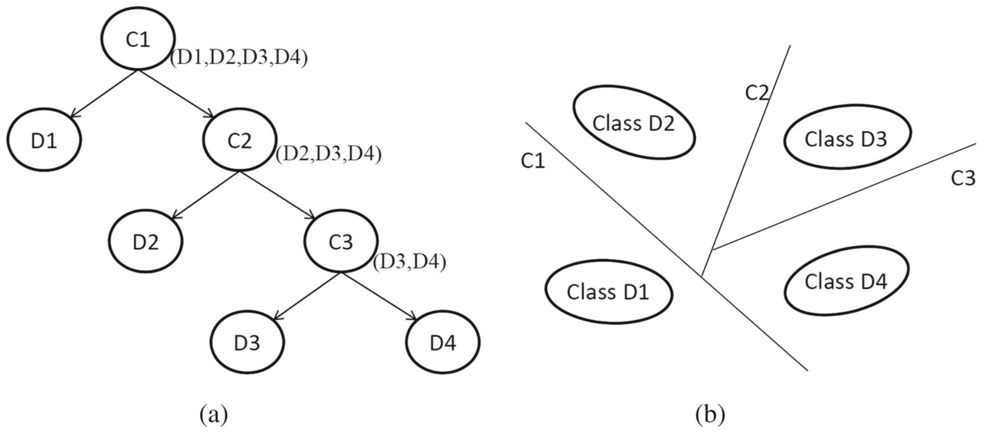


Figure 2 **a** Structure of BTSVM, **b** Generalization region by BTSVM.

two types of RDD: parallelized collections that are based on local files (like map, list, etc) and files that are stored on remote file systems (like Amazon-S3, Hadoop distributed system, etc). After RDD is constructed, the programmer can perform transformations and actions. Transformations (map, filter, etc) create new data sets from the existing RDD or the input file, while actions (toArray, length, collect, saveAsTextFile, etc) return a value after executing calculations on the RDD. It is noticeable that the transformations are the lazy operation, which means that the transformations commands are not performed immediately on the RDD; rather, actions perform the actual computation and calculate the result or save to the external storage [30].

During execution, a directed acyclic graph is generated taking all the transformation dependencies into account, whenever the users run an action on RDD. As a result, Spark can express more programs through kinds of combinations of RDD and transformations on them [31].

However, there is no general framework focusing on the data preprocessing for missing data prediction and type classification in power grid system at present. Specifically, there is no universal framework aiming at big-data-oriented fault diagnose of electric equipment. So, it is necessary to integrate the appropriate algorithms which can effectively deal with data preprocessing with big data analysis to prepare high-quality data for data analysis.

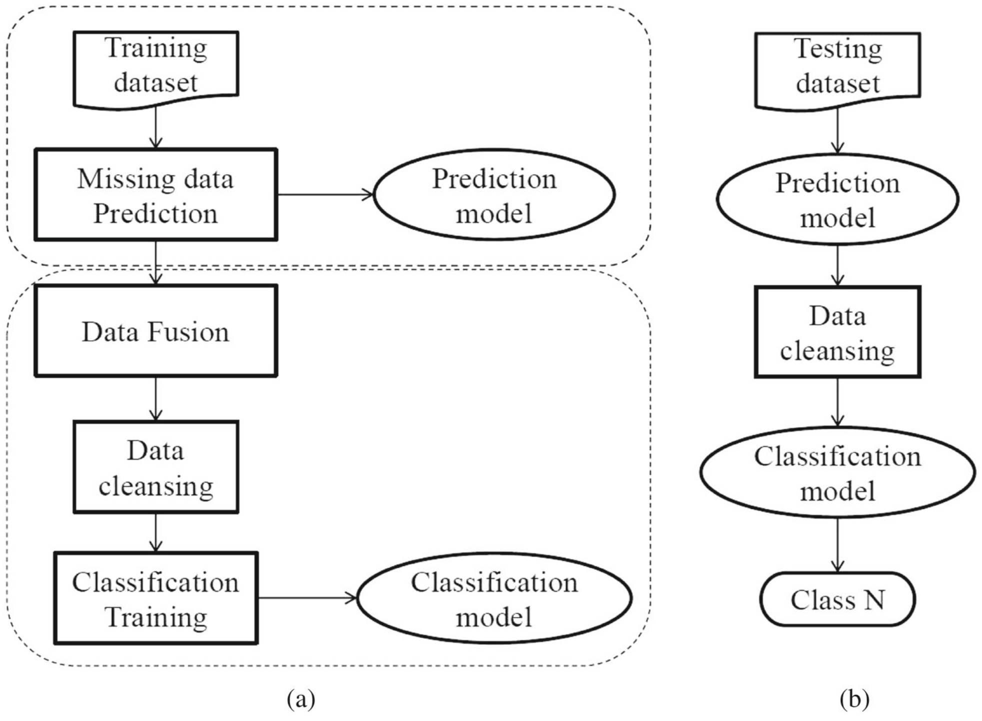


Figure 3 Overview of the data preprocessing framework.

3 The Proposed Data Preprocessing Framework

In this section, we first propose an integrated data preprocessing framework (DPF), which includes key components which mainly include missing data prediction and fault type classification parts. Then, we present detailed realization of the framework based on Spark (Fig. 3).

3.1 Overview of Data Preprocessing Framework

Figure 3a illustrates the overall structure of DPF which include two major steps, which include missing data prediction and fault type classification. The classification step can be further divided into data fusing, data cleansing and classification steps. Afterwards, based on the trained prediction model and classification model, we can classify fault type of equipment in power grid system as shown in Fig. 3b. We will discuss the above parts of the framework step by step in the following sections.

3.2 Detailed missing data prediction in DPF framework

Figure 4 illustrates the overall structure of detailed missing data prediction in DPF framework: 1) extracting features which include missing data from electronic data gathering equipment; 2) training one fundamental linear regression model LinRM and optimizing the trained model; 3) testing and comparison with the data in real world.

3.2.1 Extracting Features

For each sensor facility, its measured samples are represented as a discrete time series - observations are made at fixed time intervals and form a discrete set of values. In order to improve the effect of missing data prediction, these time series are preprocessed through normalization, which replaces nonstandard data values with corresponding values that comply with the standard to adjust values measured on different scales to a notionally common scale [18]. In each case, features $x_1, x_2, \ldots, x_M$ are extracted as follows:

$$x_k = a + \frac{(D_k - \min D_k)(b - a)}{\max D_k - \min D_k}, \qquad (12)$$

where $k = 1, 2, 3, \ldots, M$, D_k is the authentic value of the above time series in one same feature. Besides, a and b are generalized to restrict the range of values in the data set. Here, we set a=1 and b=2. Then, feature vector $X = [x_1, x_2, x_3, \ldots, x_M]$ for SVM is composed.

3.2.2 Training and Optimizing Linear Regression Model

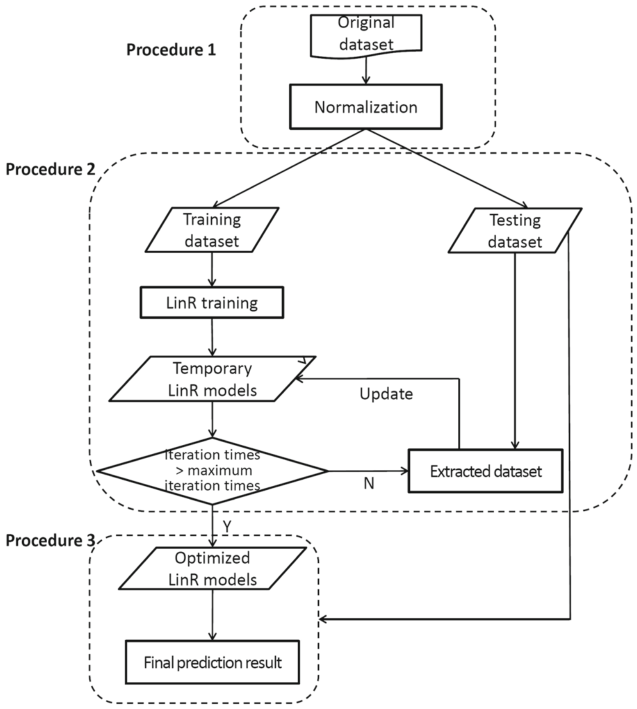


Figure 4 Flow chart of detailed missing data prediction framework.

Algorithm 1 (Procedure 4: Train Optimized Linear Regression)

Input: training data set X, preliminary LinR model $LinRMs$, sampling proportion Σ, maximum iteration number $maxIter$
Output: optimized linear regression model OLR

```
 1: function TRAINOLR($X$, $LinRMs$, $\Sigma$, $maxIter$)
 2:     for each training data set $x$ in $X$ do
 3:         for $k \rightarrow maxIter$ do
 4:             $p_0$ ← missing data prediction result of $x$ based on corresponding LinR model $LinRMs$
 5:             $errors$ ← computing the prediction error of $p_0$
 6:             $nx$ ← constructing one new data set according to formula (5)
 7:             $newX$ ← merging the new data set $nx$ with $X$
 8:         end for
 9:     end for
10:     $OLR$ ← updating the $LinRMs$ with $newX$
11:     return $OLR$
12: end function
```

The main process of training and optimizing linear regression model is shown in Algorithm 1. Based on the training data set X after normalization, we first separately train each feature of the existing data set by traditional LinR and obtain preliminary LinR models. Second, the temporary prediction results could be achieved by the acquired LinR model. Next, the most relevant samples could be extracted by comparing the mean square errors (MSE) between real original data and the predicted data with respect to Σ which is set artificially. According to formula (5), we can construct a new data set *newX*. Then, the training data set is updated by simply merging the new data set nx with the original data set. Finally, the optimized LinR (OLR) is obtained by updating the preliminary LinR model based on the newly formed data set.

Taking x_1 for example, some data are lost in x_1 during some period. First we obtain an initial $LinRM\_x_1$ model by training traditional linear regression with x_1. Next the corresponding prediction data p_0 of training data x_1 is calculated by $LinRM\_x_1$, and then the prediction errors of training data samples are calculated. The new data set nx is constructed according to the threshold Σ which is set before. The merged data set $newT$ is composed of original testing data set and nX. Finally, the new OLR is achieved by updating the $LinRM_{x1}$ based on $newT$.

3.3 Detailed Fault Type Classification in DPF Framework

The second part of the DPF framework is composed of three parts, which include data fusion, data cleansing and fault type classification. In the subsections, will discuss the three parts respectively.

3.3.1 Data Fusion

Considering there might be hidden relationship among features from different sources, we attempt to find the most relevant relation between different features before training the classification model. The Pearsons Correlation Coefficient (PCC) is a metric of the linear correlation between two data sets X and Y, which is widely used in pattern recognition, statistical analysis, and image processing [14]. The formula of computing PCC is:

$$r = r_{xy} = \frac{\sum_{i=1}^{N}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{N}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{N}(Y_i - \overline{Y})^2}} \tag{13}$$

where N is the total number of samples, X_i is the ith sample in data set X, and $\overline{X}$ is the mean value of data set X. The same holds for Y_i and $\overline{Y}$.

3.3.2 Data Cleansing

After applying the procedure described in last part, there would be new data sources to incorporate into the calculation of fault diagnosis. Consequently, there is an increase in the dimension and noise of the data set, producing side effects on the accuracy of fault pattern recognition. In this subsection, we adopt a data cleaning process based on principal component analysis (PCA). The general process of PCA and the details could be found in [32]. It is noticeable that the proportionate effect of the noise is less-the first few components achieve a higher signal-to-noise ratio. Therefore, PCA can have the effect of concentrating much of the signal into the first few principal components, while the later principal components may be dominated by noise, and so disposed of without great loss.

3.3.3 Fault Type Classification Based on BTSVM

Algorithm 2 Train BTSVM

```
Input: $TrainRDDData$: the RDD data set for training
Output: $claModels$: classification models for fault type classification
 1: function TRAINPREDICTION($dpTrain$)
 2:     step TRAIN_BTSVM ($trainRDDData$) {
 3:         {macro-class data $M_i$, macro-class data $M_j$} $\leftarrow$ dividing the multi-class data $trainRDDData$ into two macro-classes
 4:         $claModels \leftarrow$ SVM.train($M_i$, $M_j$)
 5:         // SVM: binary SVM
 6:         if $M_i$ contain more than one class
 7:         TRAIN_BTSVM($M_i$)
 8:         if $M_j$ contain more than one class
 9:         TRAIN_BTSVM($M_j$)
10:     }
11: end function
```

The main process of training BTSVM model is shown in Algorithm 2. Based on the training data set $inRDDData$, we first divide the data set into two macro-classes M_i and M_j [33]. Second, the traditional SVM for binary classification is trained based on the two macro-classes. Finally, if M_i or M_j still contains more than one class, the data set is recursively inputted into the training function. It is important to note that the model $claModels$ has $K - 1$ classifiers which has a binary tree structure as shown in Fig. 2.

3.4 Implementation of DPF Framework Based on Apache Spark

Algorithm 3 **Missing Data Prediction Algorithm on Apache Spark**

```
Input: $dpTrain$: the data path of training data set, $dpPred$: the data path of the data set for prediction, sampling proportion $\Sigma$, maximum iteration number $maxIter$
Output: $optPredModel$: optimized prediction model, $predRes$: prediction result of the data set for prediction
1:  function TRAINPREDICTION($dpTrain$)
2:      step READ ($dpTrain$) {
3:          $inRDDData \leftarrow$ SparkContext.textFile($dpTrain$)
4:          // inRDDData : RDD[ID] = [label; f1, f2, f3, ...]
5:      }
6:      step TRAIN ($inRDDData$) {
7:          $trainModel \leftarrow$ LinearRegression.train($inRDDData$)
8:      }
        $optPredModel \leftarrow$ trainOLR ($inRDDData, trainModel, \Sigma, maxIter$)
9:      return $optPredModel$
10: end function
11: function PREDICTION($dpPred, optPredModel$)
12:     step READ ($dpPred$) {
13:         $predRDDData \leftarrow$ SparkContext.textFile($dpPred$)
14:         // predRDDData : RDD[ID] = [label; f1, f2, f3, ...]
15:     }
16:     step PRED ($predRDDData, optPredModel$) {
17:         $predRes \leftarrow$predRDDData.mapPartitions($p$)
18:                 {optPredModel.predict($p.getDatapoint$)}
19:         // p : each of predRDDData ;
20:     }
21:     return $predRes$
22: end function
```

We study implementation issues for our DPF framework based on Apache Spark. Algorithm 3 shows the general process of training linear regression model and OLR on Apache Spark for prediction. First, the framework builds the initial training model after transforming the original input data into RDD. Next, an initial model called $trainModel$ is trained based on $inRDDData$. Later, depending on the Algorithm 1 in Section 3.2.2, we can obtain the optimized OLR model $optPredModel$. During the prediction step, the mapPartition function takes each instance of the $predRDDData$ called p and predicts the temporary results based on the optimized model.

Algorithm 4 **Fault Type Classification Algorithm on Apache Spark**

```
Input: $predRes1$, $predRes2$: the data set from different data sources after preprocessing, $dpCla$: the data path of the data set for classification,
Output: $claModel$: classification model , $faultTypes$: fault type classification
1:  function (trainClassification)$predRes1, predRes2$
2:      step FINDCORRELATION ($predRes1, predRes2$) {
3:          $correlationship \leftarrow$ PCC ($predRes1$, $predRes2$)
4:          // correlationship : Array[FeatureID]
5:      }
6:      step PCA ($correlationship$) {
7:          $newData \leftarrow$ constructing new training data set based on $correlationship$
8:          // newData : localized
9:          $trainRDDData \leftarrow$ PCA(parallize ($newData$))
10:     }
11:     $claModels \leftarrow$ TRAIN_BTSVM ($trainRDDData$)
12:     return $claModels$
13: end function
14: function CLASSICATION($dpCla, claModels$)
15:     step READ ($dpCla$) {
16:         $claRDDData \leftarrow$ SparkContext.textFile($dpCla$)
17:         // claRDDData : RDD[ID] = [label; f1, f2, f3, ...]
18:     }
19:     step CLAS ($claModels$) {
20:         $faultTypes \leftarrow$ claRDDData.mapPartitions($p$) {claModel.predict($p.getDatapoint$)}
21:         // p : each entity of claRDDData ;
22:     }
23:     return $faultTypes$
24: end function
```

Algorithm 4 shows the general process of training BTSVM model on Apache Spark for classification. First, in order to find hidden correlation, two different data set from different data sources are chosen as the original training data set. Next, through Pearsons Correlation Coefficient (PCC) analysis, the feature vector with the biggest coefficient in $predRes2$ is selected as the new feature vector in $predRes1$. Later, since the newly added feature vector would bring not only useful and valuable information

Table 1 Mean Square Error (MSE) OF BPNN, SVM AND LinR for Single Missing Data Prediction in x_1(H_2).

Methods	BPNN	SVM	LinR
MSE($X_1(H_2)$)	0.972	0.875	0.273

but also additive noise, we use principal component analysis (PCA) to select the principal components and reduce the noise in the $newData$. Finally, we achieve the BTSVM algorithm based on the SVM for binary classification which is provide on the Apache Spark platform. By training the BTSVM regression model, we can obtain the final classification models $claModels$, whose structure is as Fig. 2 shows. During fault type classification process, the mapPartition function inputs each instance of the $claRDDData$ called p into the classification model $claModels$ layer by layer till we get a classification result, that is to say, till we get to the leaf node on the tree structure. For comparison, we also use logistic regression in Spark as the classification algorithm. The general process of training and classification logistic regression model for classification is almost as the same as that of BTSVM model.

As Spark is implemented in Scala, which is a functional programming language that runs on Java platform, we also use the same language. In contrast to traditional languages like C, similar expressions in Scala may behave differently and introduce overheads in developing Scala programs. Thus, it is necessary to carefully design the Scala programs. For example, in our DPF framework, we use mapPartitions instead of map. The mapPartitions operation in Spark applies a function on each partition of an RDD to reduce overheads.

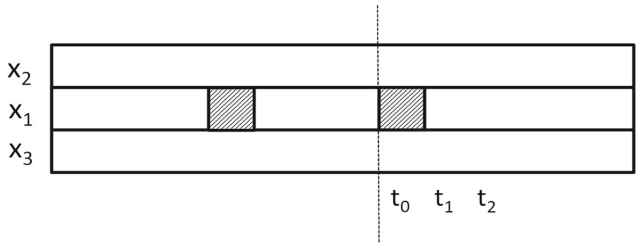


Figure 6 A practical missing data situation in transformer monitor systems. The single shaded box represents one missing data.

4 Experiments

In power grid system, the states of transformers are indicated by the amounts of contents of certain diagnostic gas, including hydrogen (H_2), methane (CH_4), ethylene (C_2H4), ethane (C_2H_6), carbon monoxide (CO), acetylene (C_2H_2) and carbon dioxide (CO_2). These seven kinds of gases are designated as the major diagnostic parameters $x_1, x_2, x_3, x_4, x_5, x_6, x_7$, respectively. To evaluate the prediction and the classification accuracy of the algorithms in Section 3 under DPF framework, first, gases samples collected from a 220kV main transformer in State Grid of China are selected as the original training data set for prediction. And we attempt to predict the missing data in x_1 (H_2). Then, the training data set which contains above seven kinds of diagnostic gases is respectively inputted into linear

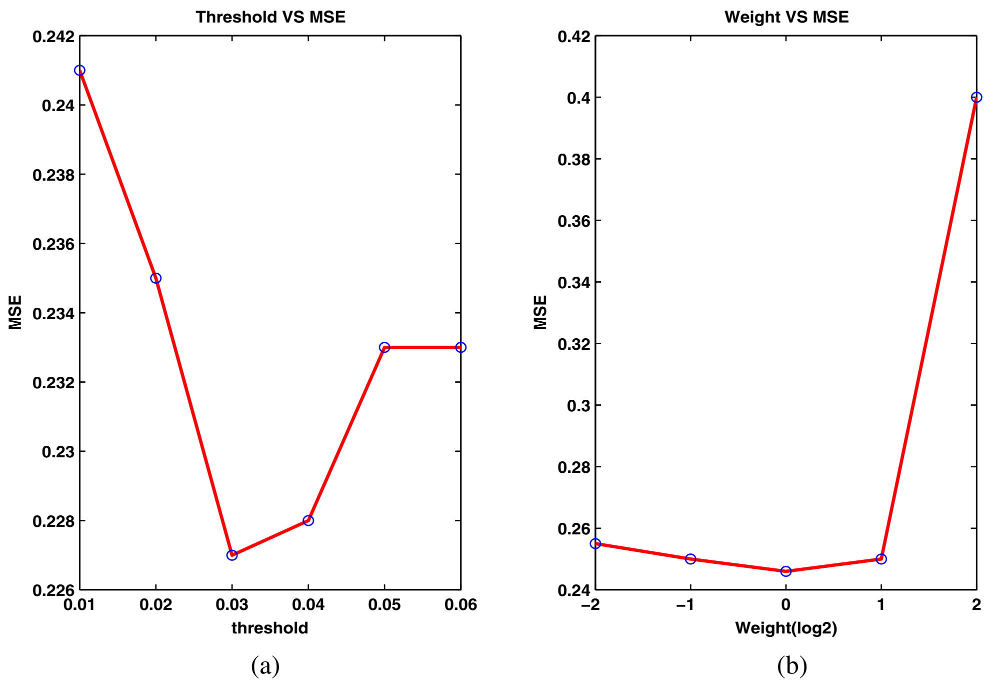


Figure 5 **a** MSE of OLR using different threshold when keeping weight unchanged; **b** MSE of OLR using different weight when keeping threshold unchanged.

Table 2 Mean Square Error (MSE) OF LinR and OLR for Single Missing Data Prediction in $x_1(H_2)$.

	H_2	CH_4	C_2H_4	C_2H_6	CO	C_2H_2	CO_2
LinR	0.273	0.021	0.345	0.16	0.1	0.12	0.19
OLR	0.226	0.019	0.39	0.164	0.12	0.113	0.17

regression (LinR) and traditional classic models like back propagation neural network (BPNN) and SVM. Later, three missing data prediction models are produced for testing. Also, we test the performance of our proposed optimized LinR (OLR) in DPF. Besides, State Grid of China Incorporation also offered us the original data including all these gases and other data such as oil temperature, the environmental humidity and load current from different monitoring systems. Using the scheme proposed in this paper, we firstly find correlation between the new features and traditional diagnostic gases from the sensor data in DGA. Furthermore, fault records selected as our training set, we compare the classification results between the existing methods and our scheme. Finally, our proposed framework DPF is also evaluated under parallel environment. All the traditional methods are conducted under Matlab version 2012b and the other methods are conducted under Apache Spark. The multi-class classification based on SVM takes the BTSVM strategy because it only trains $K - 1$ SVM classifiers which benefit the speed of training when dealing with the big data problems.

5 Results and Analysis

In this section, we will analyse the results of experiments which are setup in Section 4.

5.1 Evaluation Under Stand-Alone Computer Environment

5.1.1 Comparison Between LinR and Traditional Methods

To evaluate the performance of kinds of predictors, mean squared error (MSE) is employed to measure the average of the squares of the "errors", that is, the difference between the estimator and what is estimated [7]. Taking testing data set t_1 (H_2) for example, the MSE_{t1} is calculated as follows:

$$MSE\_t_1 = \sqrt{\frac{\sum_{i=1}^{N_{t1}} (p_1^i - t_1^i)^2}{N_{t1} - 1}} \qquad (14)$$

where t_1^i is one testing data sample and p_1^i is the predicted result of missing data and N_{t1} is the total number of t_1.

Table 1 shows the prediction result of LinR and traditional methods. It needs to be made clear that the SVM here is one traditional prediction method for predicting missing values, which is different from BTSVM in the proposed framework. Compared with BPNN and SVM, LinR shows the lowest MSE, which is 0.273. So, it is reasonable to take LinR as the fundamental training model for optimization.

5.1.2 Comparison Between LinR and OLR

To evaluate the performance of OLR, we should set reasonable values of two parameters, which are threshold λ and weight σ as shown in formula (5).

First, we keep weight unchanged and attempt to optimize the parameter λ. As shown in Fig. 5a, when λ equals 0.03, the MSE of OLR reaches to the lowest value. Similarly, we keep threshold λ unchanged and try to optimize the parameter σ. As shown in Fig. 5b, when σ equals 1, the MSE of OLR reaches to the lowest value. As a consequence, we reasonably set $\lambda = 0.03$ and $\sigma = 1$. After using the optimized and as the parameters of OLR, the MSE of OLR is 0.226, which is reduced by 17 percent.

Considering the fact that strong association exists among the seven different diagnostic gases and the practical situation shown in Fig. 6, we try to predict missing data utilizing different gases, the data set of which might be complete at corresponding time, such as x_2 (CH_4) and x_3 (C_2H_4) in Fig. 6.

Table 2 shows the prediction result of x_1 (H_2) using LinR and OLR based on the data itself and different diagnostic gases. On the one hand, the result shows that the OLR outperforms the LinR on the whole. The main reason is because the predicted value of is x_1 (H_2) more close to the real value as revealed by formula (9). On the other hand, the result also shows that OLR in the DPF is more adaptable to predict missing data by different diagnostic gases. For example, the MSE obtained using x_2 as prediction data (0.021) is even lower than that is obtained using x_1 itself (0.273). Thus, the propose optimized linear regression model obtains better performance and the proposed prediction strategy utilizing different diagnostic gases is an alternative method for prediction the missing data in power grid system.

Table 3 Classification Accuracy Between BPNN, BTSVM AND LOGR.

Methods	BPNN	BTSVM	LogR
Accuracy (%)	73	82.1	94.0

Table 4 Pearson Correlation Coefficient of Total Hydrocarbons and Other Features.

Sensor data from other sources	OL	EH	LC
Absolute value of r	0.81	0.106	0.212

5.1.3 Comparison Between LogR, BTSVM and BPNN

To evaluate the performance of classification algorithms for fault diagnose, we first compare the prediction accuracy of BPNN, BTSVM and logR. As Table 3 shows, the classification result reveals that the BTSVM and LogR obtain higher accuracy and are more adaptable to be used for classification in this case.

5.1.4 Performance of LogR and BTSVM with Principal Component Analysis

Power production management system (PMS) and dissolved gas analysis (DGA) are two most common systems in State Power Grid. To deduce whether the data set Z in PMS are correlated in data samples in DGA, we first calculate the summation Sum_X_i of each sample X_i. Then, one new feature vector $Sum\_X = [Sum\_X_1, Sum\_X_2, \ldots, Sum\_X_N]$ is composed. Next, the Pearson Correlation Coefficient r_i between each feature Z_i in Z and Sum_X is obtained based on Eq. (13). Finally, if r_i is larger than threshold ψ which is set manually, the corresponding feature Z_i is chosen as the new feature and added into the original data set. Electrical engineers usually understand that hydrocarbon gases contribute most to the transformer fault information among the dissolved gases. Hence we could figure out one diagnostic parameter x_8 for classification by calculating the Pearson coefficient r between Total Hydrocarbons (sum of all hydrocarbon gases concentration, namely CH_4, C_2H_2, C_2H_4 and C_2H_6) and other sensor data in PMS, which include oil temperature (OL), environmental humidity (EH) and load current (LC).

From Table 4, we can conclude that the oil temperature has very strong correlation with hydrocarbon gases and Fig. 7 also proves this by showing the trends of total hydrocarbons and oil temperature over time. Since oil temperature is identified as the most related data feature, we should add oil temperature as x_8 to the diagnostic features for fault classification. Next, we compare our new method and the conventional method without using oil temperature based on LogR and BTSVM.

It is shown in Table 5 that the classification accuracy of LogR degrades about 3 % when OL is added into the training data set. Similarly, the classification accuracy of BTSVM decreases about 1 %. We deduce that although the increase of the dimension of the training brings more information, it also result in much more noise, which deteriorates the performance of the classifiers. To reduce the noise brought by OL, we apply PCA to the new data set before training. Usually, the number of principal components L is chosen as small as possible while satisfying that the cumulative energy g is above a certain threshold, like 90 percent. In this case, L equals 5.

The result in Table 5 reveals that, as for LogR, the accuracy of fault diagnosis through reducing the dimensionality and noise of data set by PCA is improved about 4.7 %. Similarly, as for SVM, the accuracy is improved about 2.2 %. The main reason is because the framework DPF with PCC and PCA brings sufficient data information and reduces the noise at the same time.

Table 5 Comparison of the Classification Accuracy Between LogR and BTSVM in Our Schema Based on the Data With OL.

Accuracy of classification (%)	Data with OL	Data with OL after PCA
BTSVM	81.4	83.2
LogR	91.4	95.7

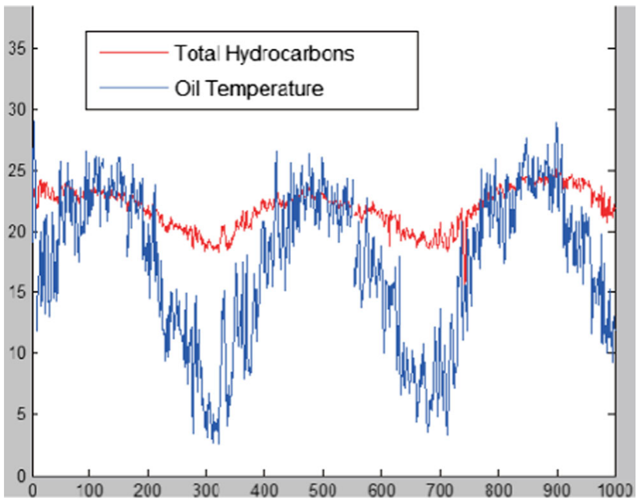


Figure 7 Trends of total hydrocarbons and oil temperature over time.

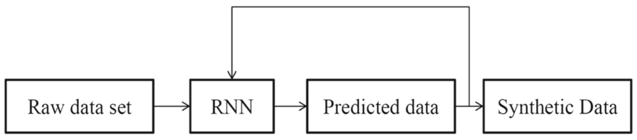


Figure 8 Process of synthesizing big data.

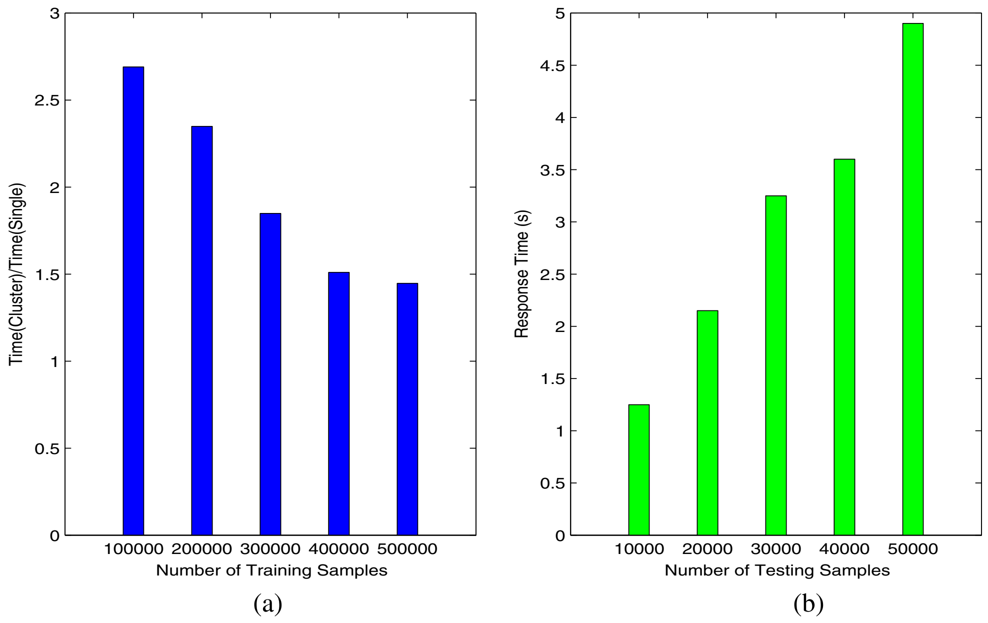


Figure 9 **a** Ratios of the time used for training the OLR and LogR in DPF framework between stand-alone computer and cluster under different number of training samples; **b** Response time under different number of testing samples.

5.2 Evaluation Under Parallel Environment

Since the superior performance of the framework DPF is achieved, which means that the framework can effectively preprocess the time series data in our case, we attempt to evaluate performance of DPF when it comes to dealing with big data case. Parallel computing experiments are carried out in a cluster of five working machines. The working nodes are virtual machines (VM) and each of them has two cores from an Intel 2400 CPU and 4G memory. The operating system for the cluster is CentOS 7, while the version of Apache Spark platform is 1.4.0 and the Hadoop platform is version 2.6.0.

The big data sets are synthesized based on the raw data set, which originates from DAG and PMS systems. As Fig. 8 shows, the raw data is first inputted in recurrent neural network (RNN) [34], which is an effective nonlinear autoregressive moving average model, to produce new time series data. Then, the newly predicted time series could also be utilized as the input data of RNN. The final big data sets are synthesized by repeatedly running the last two steps. In our case, the total sample number of the synthetic data varies from 100 thousand to 500 thousand.

To make the experimental results clearer, we take the ratios of the time used for training the OLR and LogR in DPF framework between stand-alone computer and cluster as the metric. Figure 9a shows the ratios under different number of training samples. The experimental result shows that the ratio decreases with the increase of the number of training sample. However, the decrease trend tends to be subsiding due to the rapidly increasing demand of computing memory and the limitation of our experimental environment. In fact, when the number of training samples reaches 550 thousand, the memory of our cluster nearly runs out. Thus, more time will be spent on communication between working nodes and reading and writing operations between memory and hard disks. Although the experimental environment limits further accurate analysis, we could also deduce that the training time would be largely reduced if the memory and the number of working nodes are reasonably increased.

After the prediction models are trained, we also conduct the experiments on the response time of DPF under different number of testing samples, which is shown in Fig. 9b. The plot reveals that the response time is less than five seconds when the number of testing samples is as high as 50 thousand. As we conduct the experiments through offline data analysis in the specific scenario of power grid where evaluation decisions are usually made several months before a predicted failure, the response time is in a reasonable range.

6 Conclusion

In this paper, we propose an improved data preprocessing framework DPF on Apache Spark for missing data prediction and fault type diagnose incorporating optimized LinR, data fusion and data cleansing to improve the quality of raw data generated by State Grid of China. As expected from our improved approaches, our framework DPF exhibits higher precision in terms of lower MSE than classic SVM and NN methods, e.g., the predictor under DPF combining the proposed strategy utilizing different diagnostic gases with OLR obtains the lowest MSE 0.019. Besides, our framework further reduces the data set's dimensionality as well as noise in the data set to form a concise high quality data set as input to classifiers. As a consequence, DPF also shows higher classification accuracy when the training data is processed through data fusion and data cleansing by PCC and PCA. We also deduce that the training time would be largely reduced if the memory and the number of working nodes are reasonably increased, although the experimental environment limits further accurate analysis. Due to the nature of linear regression, logistic regression and SVM, our framework keeps good robustness, leading to a strong candidate for online missing data prediction and fault diagnose in power grid equipment monitoring and evaluation. Though the DPF framework works well with missing data prediction and fault diagnose of power transformers, there are rooms for improvement, such as how to collect more relevant operating data of transformers, which need to be studied in the future.

Acknowledgments This paper is sponsored in part by the National High Technology and Research Development Program of China (863 Program, 2015AA050204), State Grid Science and Technology Project (520626140020, 14H100000552, SGCQDK00PJJS1400020), State Grid Corporation of China, the National Research Foundation Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program, and the National Natural Science Foundation of China (No.61373032).

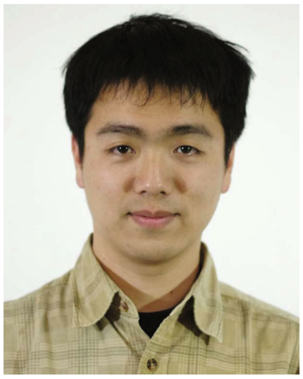
References

1. Batini, C., Cappiello, C., Francalanci, C., & Maurino, A. (2009). Methodologies for data quality assessment and improvement. *ACM Computing Surveys*, *41*(3), 1–52.
2. Niu, J., Gao, Y., Qiu, M., & Ming, Z. (2012). Selecting proper wireless network interfaces for user experience enhancement with guaranteed probability. *Journal of Parallel and Distributed Computing*, *72*(12), 1565–1575. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731512002134.
3. Li, Y., Dai, W., Ming, Z., & Qiu, M. (2015). Privacy protection for preventing data over-collection in smart city. *IEEE Transactions on Computers*, *PP*(99), 1–1.
4. Lee, K., Kung, S.-Y., & Verma, N. (2012). Low-energy formulations of support vector machine kernel functions for biomedical sensor applications. *Journal of Signal Processing Systems (JSPS)*, *69*(3), 339–349. [Online]. Available: doi:10.1007/s11265-012-0672-8.
5. Zliobaite, I., & Gabrys, B. (2014). Adaptive preprocessing for streaming data. *IEEE Transactions on Knowledge and Data Engineering*, *26*(2), 309–321.
6. Davis, J.J., & Clark, A.J. (2011). Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, *30*(6–7), 353–375. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404811000691.
7. Khalighi, S., Pak, F., Tirdad, P., & Nunes, U. (2015). Iris recognition using robust localization and nonsubsampled contourlet based features. *Journal of Signal Processing Systems (JSPS)*, *81*(1), 111–128. [Online]. Available: doi:10.1007/s11265-014-0911-2.
8. Qiu, M., Ming, Z., Li, J., Liu, J., Quan, G., & Zhu, Y. (2013). Informer homed routing fault tolerance mechanism for wireless sensor networks. *Journal of Systems Architecture*, *59*(4–5), 260–270. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762113000040.
9. Ma, H., King, I., & Lyu, M.R. (2007). Effective missing data prediction for collaborative filtering. In *Inproceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 39–46). Amsterdam: ACM.
10. Nogueira, R., Vieira, S., & Sousa, J. (2005). The prediction of bankruptcy using fuzzy classifiers. In *2005 ICSC Congress on Computational Intelligence Methods and Applications* (p. 6).
11. Lei, K.S., & Wan, F. (2010). Pre-processing for missing data: A hybrid approach to air pollution prediction in macau. In *2010 IEEE International Conference on Automation and Logistics (ICAL)* (Vol. 16–20, pp. 418–422).
12. Tian, F., Sun, J., & Shao, S. (2013). Wavelet threshold de-noising applications in avionics test data processing. In *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC)* (Vol. 21–23, pp. 667–671).
13. Wei, X., Xiao, B., Zhang, Q., & Liu, R. (2011). A rigid structure matching-based noise data processing approach for human motion capture. In *2011 Workshop on Digital Media and Digital Content Management (DMDCM)* (Vol. 15–16, pp. 91–96).
14. da Silva, I., & Adeodato, P. (2011). Pca and gaussian noise in mlp neural network training improve generalization in problems with small and unbalanced data sets. In *The 2011 International Joint Conference on Neural Networks (IJCNN)* (pp. 2664–2669).
15. Yu, L., Wang, S., & Lai, K. (2006). An integrated data preparation scheme for neural network data analysis. *IEEE Transactions on Knowledge and Data Engineering*, *18*(2), 217–230.
16. Atasu, K. (2015). Feature-rich regular expression matching accelerator for text analytics. *Journal of Signal Processing Systems (JSPS)*, 1–17. [Online]. Available: doi:10.1007/s11265-015-1052-y.
17. Karthikeyan, P., Amudhavel, J., Abraham, A., Sathian, D., Raghav, R.S., & Dhavachelvan, P. (2015). A comprehensive survey on variants and its extensions of big data in cloud environment. In *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering and Technology (ICARCSET 2015)* (pp. 1–5). Unnao: ACM.
18. Morchen, F., & Ultsch, A. (2005). Optimizing time series discretization for knowledge discovery. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (pp. 660–665). Chicago: ACM.
19. Shi, W., Zhu, Y., Zhang, J., Tao, X., Sheng, G., Lian, Y., Wang, G., & Chen, Y. (2015). Improving power grid monitoring data quality: An efficient machine learning framework for missing data prediction. In *IEEE 17th International Conference on High Performance Computing and Communications, 2015* (pp. 417–422). IEEE Computer Society.
20. Zhang, J., Zhu, Y., Shi, W., Sheng, G., & Chen, Y. (2015). An improved machine learning scheme for data-driven fault diagnosis of power grid equipment. In *The 2015 IEEE International Symposium on Smart Data* (pp. 1737–1742). IEEE Computer Society.
21. Lu, Z., & Hui, Y. (2003). L 1 linear interpolator for missing values in time series. *Annals of the Institute of Statistical Mathematics*, *55*(1), 197–216. [Online]. Available: doi:10.1007/BF02530494.

22. Hong, S.T., & Chang, J.W. (2011). A new data filtering scheme based on statistical data analysis for monitoring systems in wireless sensor networks. In *Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications* (pp. 635–640). IEEE Computer Society.
23. Grunwald, P. (2007). Linear regression. In *The Minimum Description Length Principle* (pp. 335–368). MIT Press. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6282057.
24. Trevor, H., Robert, T., & Jerome, F. (2001). *The elements of statistical learning: data mining, inference and prediction* (Vol. 1, pp. 371–406). New York: Springer.
25. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.
26. Abe, S. (2003). Analysis of multiclass support vector machines. *Thyroid*, *21*(3), 3772.
27. Lin, C.-Y., Tsai, C.-H., Lee, C.-P., & Lin, C.-J. (2014). Large-scale logistic regression and linear support vector machines using spark. In *IEEE International Conference on Big Data (Big Data), 2014* (pp. 519–528). IEEE.
28. Solaimani, M., Iftekhar, M., Khan, L., Thuraisingham, B., & Ingram, J.B. (2014). Spark-based anomaly detection over multi-source vmware performance data in real-time. In *IEEE Symposium on Computational Intelligence in Cyber Security (CICS), 2014* (pp. 1–8). IEEE.
29. Harnie, D., Vapirev, A.E., Wegner, J.K., Gedich, A., Steijaert, M., Wuyts, R., & De Meuter, W. (2015). Scaling machine learning for target prediction in drug discovery using apache spark. In *Proceedings of the 15th IEEE/ACM International Symposium on Cluster Cloud and Grid Computing*.
30. Shanahan, J.G., & Dai, L. (2015). Large scale distributed data science using apache spark. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2323–2324). Sydney: ACM.
31. Stoica, I. (2014). Conquering big data with spark and bdas. *SIGMETRICS Perform Evaluation Review*, *42*(1), 193–193.
32. Jolliffe, I. (2014). Principal component analysis. In *Wiley StatsRef: Statistics Reference Online*. Wiley. [Online]. Available: doi:10.1002/9781118445112.stat06472.
33. Sun, G., Wang, Z., & Wang, M. (2008). A new multi-classification method based on binary tree support vector machine. In *3rd International Conference on Innovative Computing Information and Control, 2008. ICICIC '08* (p. 77).
34. Dorffner, G. (1996). Neural networks for time series processing. *Neural Network World*, *6*, 447–468.

Weiwei Shi received his Bachelor and his M.Sc degree in College of Opto-Electronic Engineering from Nanjing University of Posts and Telecommunications, China, in 2010 and 2013, respectively. He is pursuing his Ph.D degree in the School of Electronic Information and Electrical Engineering at Shanghai Jiao Tong University. His current research interests are focused on data mining and big data.

Yongxin Zhu is an Associate Professor with the School of Microelectronics, Shanghai Jiao Tong University, China. He is also a visiting Associate Professor with National University of Singapore. He is a senior member of IEEE and China Computer Federation. He received his B.Eng. in EE from Hefei University of Technology, and M. Eng. in CS from Shanghai Jiao Tong University in 1991 and 1994 respectively. He received his Ph.D. in CS from National University of Singapore in 2001. His research interest is in computer architectures, embedded systems and system-on-chip. He has authored and co-authored over 90 English journal and conference papers and 30 Chinese journal papers. He has 18 China patents approved.



Tian Huang received his PhD degree from School of Microelectronics at Shanghai Jiao Tong University in March 2016. His main research interest is Data Mining for time series, including time series big data indexing, anomaly detecting, computer architecture for time series data mining and statistical models for time series data. He has published 2 SCI journal and 15 EI conference papers. He also has rich experience on software and hardware co-designing.

Gehao Sheng was born in Hunan, China. He received his B.E., M.S. and Ph. D degree in electric power system and automation from Huazhong University of Science and Technology, Wuhan, China, in 1996, 1999 and 2003, respectively. And from July, 2003 to September, 2005, he worked in Post-Doctoral Researcher in Department of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. Currently, he is an associate professor in Department of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests is in the condition monitoring of power equipment.

Yong Lian received the B.Sc. degree from the College of Economics and Management, Shanghai Jiao Tong University, Shanghai, China, in 1984, and the Ph.D. degree from the Department of Electrical Engineering in National University of Singapore (NUS), Singapore, in 1994. Dr. Lian's research interests include biomedical circuits and systems and signal processing. He has received many awards including IEEE Circuits and Systems Society's Guillemin-Cauer Award (1996), IEEE Communications Society Multimedia Communications Best Paper Award (2008), Institution of Engineers Singapore Prestigious Engineering Achievement Award (2011), Hua Yuan Association/Tan Kah Kee International Society Outstanding Contribution Award (2013), Chen-Ning Franklin Yang Award in Science and Technology for New Immigrant (2014), and Design Contest Award in 20th International Symposium on Low Power Electronics and Design (ISLPED2015). Under his guidance, his students received many awards including the Best Student Paper Award in ICME 2007, winner of 47th DAC/ISSCC Student Design Contest in 2010, Best Design Award in A-SSCC 2013 Student Design Contest.

Dr. Lian is the President-Elect of the IEEE Circuits and Systems (CAS) Society, Steering Committee Member of the IEEE Transactions on Biomedical Circuits and Systems. He was the Editor-in-Chief of the IEEE Transactions on Circuits and Systems Part II: Express Briefs for two terms from 2010 to 2013. He was the Guest Editor for eight special issues in IEEE Transactions on Circuits and Systems-Part I: Regular Papers, IEEE Transactions on Biomedical Circuits and Systems, and Journal of Circuits, Systems Signal Processing. He was the Vice President for Publications of the IEEE CAS Society from 2013 to 2015, Vice President for the Asia Pacific Region of the IEEE CAS Society from 2007 to 2008, AdComm Member of the IEEE Biometrics Council from 2008 to 2009, CAS Society Representative to the BioTechnology Council from 2007 to 2009, Chair of the BioCAS Technical Committee of the IEEE CAS Society from 2007 to 2009, Chair of DSP Technical Committee of the IEEE CAS Society from 2010 to 2011, Member of the IEEE Medal for Innovations in Healthcare Technology Committee from 2010 to 2012, and a Distinguished Lecturer of the IEEE CAS Society from 2004 to 2005. He is the Founder of the International Conference on Green Circuits and Systems, the Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics, and the IEEE Biomedical Circuits and Systems Conference. He is a Fellow of the Academy of Engineering Singapore.

Guoxing Wang received his Ph.D. in electrical engineering from the University of California at Santa Cruz, US, in 2006.

In 2007-2009, he joined the Second Sight Medical Products, Sylmar California, where he designed the integrated circuits chip that went into the eyes of patients to restore vision. Currently, he is an Associate Professor in the School of Microelectronics, Shanghai Jiao Tong University, Shanghai, China. He has published more than sixty peer-reviewed journal and conference papers. His current research interests include wireless technologies and biomedical electronics.

Dr. Wang is currently a senior member of the IEEE and a member of the IEEE Biomedical Circuits Systems Technical Committee (BioCAS). He serves as Deputy Editor in Chief for IEEE Transactions on Biomedical Circuits and Systems (TBioCAS) (2016-2017). He serves as the technical program chair for IEEE Conference on Biomedical Circuits and Systems in 2016. He served as an Associate Editor for IEEE Transactions on Circuits and Systems II, Guest Editor for IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), Guest Editor for IEEE Transactions on Biomedical Circuits and Systems. He was the local chair for the first IEEE Green Circuits and Systems (ICGCS) and for the second Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PrimeAsia).

Yufeng Chen is a senior engineer. He received his Bachelor degree from Shanghai Jiao Tong University in 1992. Now, he is the director of Evaluation Center.