

# Short-term load forecasting with clustering–regression model in distributed cluster

Jingsheng Lei<sup>1</sup> · Ting Jin<sup>2</sup> · Jiawei Hao<sup>1</sup>  · Fengyong Li<sup>1</sup>

Received: 12 June 2017 / Revised: 16 August 2017 / Accepted: 14 September 2017 / Published online: 22 September 2017  
© Springer Science+Business Media, LLC 2017

**Abstract** This paper tackles a new challenge in power big data: how to improve the precision of short-term load forecasting with large-scale data set. The proposed load forecasting method is based on Spark platform and “clustering–regression” model, which is implemented by Apache Spark machine learning library (MLlib). Proposed scheme firstly clustering the users with different electrical attributes and then obtains the “load characteristic curve of each cluster”, which represents the features of various types of users and is considered as the properties of a regional total load. Furthermore, the “clustering–regression” model is used to forecast the power load of the certain region. Extensive experiments show that the proposed scheme can predict reasonably the short-term power load and has excellent robustness. Comparing with the single-alone model, the proposed method has a higher efficiency in dealing with large-scale data set and can be effectively applied to the power load forecasting.

**Keywords** Distributed cluster · Short-term load forecasting · Clustering–regression model · Load characteristic curve

---

✉ Jiawei Hao  
haojiaweiok@163.com

Jingsheng Lei  
jshlei@126.com

Ting Jin  
tingj@fudan.edu.cn

Fengyong Li  
fyli@shiep.edu.cn

<sup>1</sup> College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, People’s Republic of China

<sup>2</sup> School of Information Science and Technology, Hainan University, Haikou 570228, People’s Republic of China

## 1 Introduction

With the steady develop of economy and society, building the global energy Internet become more and more important [1,2]. The research of electrical power big data is one of the important sub-topics of smart grid technology. With the mining for the data from Supervisory Control and Data Acquisition (SCADA) system and Energy Management System (EMS), ones can obtain the efficient decision-making and the dispatching control of power system. As one of the most important concerns of the power industry staff, power load forecasting focus on how to extract the data related to load forecasting from the huge business flow, information flow and data flow of the smart grid and analyze these data in the environment of power big data [3,4].

Existing studies have reported common prediction methods, such as grey theory [5], fuzzy theory [6,7], time-series theory [8], artificial neural network(ANN) [9,10], wavelet theory [11], decision tree [12], support vector machine(SVM) [11,13], etc., these methods provide a variety of effective ways for power load forecasting. However, most of these methods are based on single-alone machine load forecasting model, which only adapt good historical data and small dataset. As the amount of data and dimensions increasing, methods based on single-alone machine have many shortcomings, such as over-fitting, local optimization, insufficient memory to process the large-scale dataset. Therefore, the new data processing architecture with big data and distributed computing platform is becoming more and more urgent.

With new research methods based on Apache Spark emerging [14,15], more sophisticated methods have been developed. By studying the allocation of memory in distributed cluster, an automatic algorithm with memory usage decision is proposed, which has a good application effect

[16]. Both typical clustering algorithm (k-means) and association analysis algorithm (Apriori) are deployed and tested on the distributed cluster and verified more efficient than single-alone machine on large-scale data sets [17–19], improved support vector machine algorithm, Spark support vector machine (SP-SVM), suitable for distributed environment [20], and the parameters in SVM of distributed cluster is optimized [21]. In addition, the Boosting method is used to strengthen the weak learning process component-wise linear least squares (CWLLS), and the L2-Boosting algorithm in the distributed cluster is proposed. It is used for load forecasting, results verified the validity of L2-Boosting model [22]. Short term load forecasting is usually used to arrange the start and stop of the generator, adjust the power generation ratio of the hydropower unit and thermal power unit, make the load distribution more economical, and arrange the maintenance plan of the equipment; mid-long term load forecasting is generally used for the planning and design of regional power grids [23, 24], as well as the reconstruction and capacity increase of power plants, such as increasing the generating sets of power plants.

This paper is motivated by short term load forecasting problem and makes another attempt to tackle the following challenge: short term load forecasting with power big data. We present a solution to this problem by “clustering–regression” model in distributed cluster and implemented by Apache Spark machine learning library (MLlib). The proposed scheme can predict reasonably the short-term power load and has excellent robustness, so, it is significantly superior to existing single-alone machine load forecasting model and or directly regression load forecasting model. The main contributions of this paper are as follows:

- This paper tackles short term load forecasting problem by “clustering–regression” model with distributed cluster. We emphasized that proposed scheme is rather different from existing work due to the application of “load characteristic curves of each cluster”.
- Motivated by the “load characteristic curves of each cluster”, we design “clustering–regression” short term load forecasting model. Since the “load characteristic curves of each cluster” represents the electrical features of various types of power users and is considered as the properties of a regional total load, the users’ electrical features are fully reflected and the proposed short term load forecasting model’s performance has greatly improved.
- The “clustering–regression” load forecasting model is realized by using Apache Spark machine learning library [25, 26]. We perform comprehensive performance test with “clustering–regression” model. The experimental results demonstrate that our proposed model shows a significant advantage over the directly regression model and single-alone model.

The rest of this paper is organized as follows. Section 2 describes the core technologies of Apache Spark and brief introduction of power load forecasting. In Sect. 3, we provide the detailed procedures of constructing “clustering–regression” model. The corresponding experimental results and discussions are presented in Sect. 4. Finally, Sect. 5 concludes this paper.

## 2 Related works

### 2.1 Apache Spark and MLlib

Apache Spark is a distributed cluster computing framework, which is developed in the AMPLab at UC Berkeley [27] and focuses on simplifying the design of parallel programs on cluster. In a cluster, a task scheduling server with a high memory is termed as a *Driver*, while a number of high-disk-stored task computing machines are denoted as *Workers*. In the calculation, the Driver accepts user’s commands and divides the tasks into each Worker to execute; the Worker can extract data from the Hadoop Distributed File System (HDFS) or other distributed file system. The data processed by Worker is stored in memory and then returned to the Driver to get the final result. Spark inherited MapReduce linear scalability and fault tolerance, while data processing has improved [28]. The MapReduce must be enforced to perform steps Map and Reduce sequentially; Spark can send intermediate results directly to the next step of the job through a directed acyclic graph (DAG) operator, rather than being stored in HDFS as MapReduce.

Resilient distributed datasets (RDD) is the core of Spark, has the following characteristics:

- RDD represents the data set that has been partitioned and can be operated in parallel. The organization, operation, scheduling and error recovery of the calculation task in Spark are carried out by RDD as the unit.
- RDD can be a collection of dataset that consists of data distributed on the Worker nodes.
- RDD has a fault tolerance mechanism. The dependencies between the parent RDD and the child RDD are stored in the lineage. When the data is lost or damaged during the RDD processing, the recovery can be recalculated according to the lineage.

MLlib is Apache Spark’s scalable machine learning library. Its goal is to make practical machine learning scalable and easy. It provides tools including: common machine learning algorithms, such as classification, regression, clustering, collaborative filtering, feature extraction, transformation and dimensionality reduction. In this paper, the clustering and regression correlation algorithm in MLlib is used to establish the “clustering–regression” short-term load forecasting model.

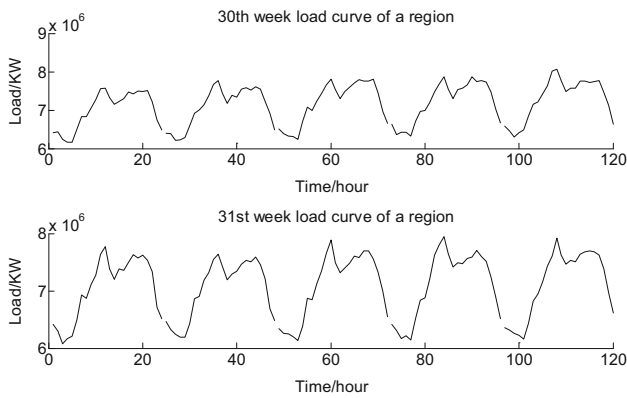


Fig. 1 Weekly load characteristic curve

### 2.2 Power load forecasting

Electrical load data has a clear periodicity, for two weeks in a region of the working load curve as shown in Fig. 1.

Figure 1 shows that the load curves of two adjacent working days in the same week are very similar. The load curves for two adjacent weeks of the same workday type (for example, *Tuesday of 30th week and Tuesday of 31st week*) are highly similar.

Let  $Load(m)$  denotes the real load of the day  $m$ ,  $Load^*(m)$  denotes the forecasting load of the day  $m$ ,  $Data(m)$  denotes the relating data of corresponding day’s power load. There are five types of  $m$ : “*Mon., Tue., Wed., Thu., Fri.*”. The purpose of load forecasting is using the existing data to train model and construct mapping  $\mathfrak{R}$  while aiming the  $Load^*(m)$ , which depended on Eq. (1), as close as possible to the actual  $Load(m)$  [29].

$$Load^*(m) = \begin{cases} \mathfrak{R} \left( \begin{matrix} Data(m-6), Data(m-7) \\ Data(m-14) \end{matrix} \right), & m \in Mon. \\ \mathfrak{R} \left( \begin{matrix} Data(m-7), Data(m-8) \\ Data(m-14) \end{matrix} \right), & m \in Fri. \\ \mathfrak{R} \left( \begin{matrix} Data(m-6), Data(m-7) \\ Data(m-8), Data(m-14) \end{matrix} \right), & else \end{cases} \quad (1)$$

### 3 Short-term load forecasting based on “clustering–regression” model

The proposed “clustering–regression” model comprises of two parts: clustering stage and regression stage. In clustering stage, according to users’ electrical features, we clustering the users and get the “load characteristic curves of each cluster”. These curves are used as a factor of the regional total load. In regression stage, combining with the corresponding day’s load, we gather relating days’ “load characteristic curve of each cluster” to train the model. The detail of proposed model is shown in **Algorithm 1**.

Algorithm1	"clustering-regression" model
2	Import data and preprocess data
3	Data clustering after processing
4	Get load characteristic curve of each cluster.
5	loop:
6	Training model
7	ifuse the model then
8	Regression
9	Else
10	Reselect data and Re-training model
11	Goto loop
12	end if
13	Analysis the result

### 3.1 “Clustering” stage

#### 3.1.1 Data preprocessing

The “user grouping raw data” and “electrical power load data” are provided by a power company in western China from March to August of 2016. The original user grouping data contains 11 attributes in total, which is the electric characters of users. The line number of “original user grouping data” is more than forty thousand, some of them are shown in Table 1. Since the existing classification is not accurate enough and meticulous, the original attributes is transformed to get five physical quantities to characterize the user’s electrical features, which includes *maximum power, load ratio, peak to total ratio, flat to total ratio and valley to total ratio* [30], where

$$load\ ratio = \frac{total\ load}{24 * 1000 * maximum\ power} \quad (2)$$

$$peak\ to\ total\ ratio = \frac{peak\ load}{total\ load} \quad (3)$$

$$flat\ to\ total\ ratio = \frac{flat\ load}{total\ load} \quad (4)$$

$$valley\ to\ total\ ratio = \frac{valley\ load}{total\ load} \quad (5)$$

The raw data is normalized by above mentioned ratio method and the ratio is basically kept between 0 and 1. Preprocessing procedure makes the electrical features is more obvious and facilitate for the program processing.

#### 3.1.2 Data clustering

Clustering the processed data with the electrical features above, the users with similar electrical features will in the same cluster. In our proposed model, the Spark k-means

**Table 1** User grouping raw data

No.	Category	Time	Total (kW)	Peak (kW)	flat (kW)	valley (kW)	Reactive power (kvar)	Max power (kW)	Rate
01*****02401	No industry classification	2011/01/10	14016.8	4285.72	6400.11	3330.97	5217.62	1.0696	100
01*****02401	No industry classification	2011/01/10	14016.78	4127.32	6873.75	3015.7	7215.01	0.7362	30
01*****02401	No industry classification	2011/01/10	14015.87	4776.14	6856.9	2382.83	3399.24	1.9922	60
01*****02401	No industry classification	2011/01/10	14015.5	3842.54	5834.39	4338.57	2014.84	1.6035	100
01*****03101	Large industry	2011/01/10	14015.49	4032.56	6330.95	3651.98	2924.92	1.4541	80
99*****49701	Large industry	2011/01/10	14015.16	3577.5	5937.79	4499.87	4756	1.2172	100
99*****6501	non-industrial power	2011/01/10	14014.46	5089.58	7402.49	1522.38	3575.17	0.6034	80
99*****49701	Large industry	2011/01/10	14014.19	4910.08	7414.79	1689.32	6353.42	1.262	50

clustering algorithm is used in clustering stage, to build “clustering–regression” model. The classical K-means algorithm is shown in **Algorithm 2** [31]. Here, “*C*” presents the number of clusters. In MLlib, the core step of implementing

Algorithm 2 k-means

**Algorithm K-means**

1	Select <i>C</i> points as the initial centroids(center point)
2	Repeat
3	Form <i>C</i> clusters by assigning all points to the closest centroid
4	Recomputed the centroid of each cluster.
5	Until the centroids don't change

the k-means algorithm as follows [32]:

- Generating the centroid of clustering; there are two ways to generate clustering centroid: “Random selection of sample points as initial clustering centroid” and “using k-means++ method to select the optimal clustering centroid”. The latter option is adopted in the proposed model. We realized this part of function by the methods *initRandom* and *initKmeansParallel* of *runAlgorithm* in the class of *KMeans*.
- Iteratively calculates the centroid of the samples. Firstly, determine the centroid class of samples. Secondly, use the aggregate function to count the sum of the values that samples belonging to each centroid and the number of samples belonging to each centroid. Thirdly, select the latest centroid through judging whether the centroid has changed.

Notably, the clustering model k-means in the MLlib defines a method which quickly determinant the distance. The method is called “quick judgment” due to the distance formula *lowerBoundOfSqDist*.

For example, the centroid is  $(a_1, b_1)$ , the sample point is  $(a_2, b_2)$ , *lowerBoundOfSqDist* is calculated as follows:

$$\begin{aligned} \text{lower Bound Of Sq Dist} &= \left( \sqrt{a_1^2 + b_1^2} - \sqrt{a_2^2 + b_2^2} \right)^2 \\ &= a_1^2 + b_1^2 + a_2^2 + b_2^2 \\ &\quad - 2\sqrt{(a_1^2 + b_1^2)(a_2^2 + b_2^2)} \end{aligned} \quad (6)$$

And the *Euclidean* distance is calculated as:

$$\begin{aligned} \text{Euclidean Dist} &= \left( a_1^2 + b_1^2 \right)^2 + \left( a_2^2 + b_2^2 \right)^2 \\ &= a_1^2 + b_1^2 + a_2^2 + b_2^2 - 2(a_1a_2 + b_1b_2) \end{aligned} \quad (7)$$

Obviously,  $\text{lowerBoundOfSqDist} \leq \text{EuclideanDist}$ . Therefore, when iterative computing the centroid, the *lowerBoundOfSqDist* is calculated firstly, that is the  $L_2$  norm of centroid and sample point. We can analysis two cases:

- (1) If the *lowerBoundOfSqDist* is not less than the minimum distance *bestDistance*, then to save the operation time, *EuclideanDist* will be not calculated.
- (2) If the *lowerBoundOfSqDist* is less than the minimum distance *bestDistance*, then *fastSquaredDistance* will be used to quickly calculate the distance.

Notably, the calculate precision of *fastSquaredDistance* method is compared with the pre-set value “*precisionBound*”. The approximate formula is used to calculate the *Euclidean* distance if the precision is satisfied, otherwise, the original formula is used.

	Cluster 1 $n_1$ Users	Cluster 2 $n_2$ Users	...	Cluster C $n_c$ Users
	$\frac{1}{n_1} \sum_{k^{(1)}=1}^{n_1} load(k^{(1)}, j)$	$\frac{1}{n_2} \sum_{k^{(2)}=1}^{n_2} load(k^{(2)}, j)$		$\frac{1}{n_c} \sum_{k^{(c)}=1}^{n_c} load(k^{(c)}, j)$
time \ cluster	cluster_1	cluster_2	...	cluster_C
0:00	339275.04	178705.7	...	357903.99
0:15	332491.85	175132.8	...	350748.36
0:30	328078.4	172808.11	...	346092.57
0:45	325105.25	171242.07	...	342956.17
⋮	⋮	⋮	⋮	⋮
23:00	368112.93	193895.42	...	388325.32
23:15	354842.4	186905.47	...	374326.13
23:30	341751.5	180010.12	...	360516.43
23:45	331353.82	174533.37	...	349547.84

Fig. 2 Accumulate the load of each measuring point (96 points)

### 3.1.3 Building the load characteristic curve of each cluster

In this paper, we perform several experiments and comparison and train the parameter “C” as 20. The definition of “load characteristic curve of each cluster” is given as Eq. (8). For the “ $n_i$ ” users in the cluster “ $i$ ”, the mean load of each metering point (total of 96 points) is calculated. An example is given in Fig. 2.

$$\begin{aligned}
 \mathbf{Load}(i) &= [l_1^{(i)}, l_2^{(i)}, \dots, l_j^{(i)}, \dots, l_{96}^{(i)}]^T \\
 l_j^{(i)} &= \frac{1}{n_i} \sum_{k^{(i)}=1}^{n_i} load(k^{(i)}, j) \\
 i &\in [1, C], j \in [1, 96]
 \end{aligned}
 \tag{8}$$

Here,  $\mathbf{Load}(i)$  is the load characteristic curve of cluster “ $i$ ”;  $l_j^{(i)}$  is the mean load of cluster “ $i$ ” at the time of “ $j$ ”;  $load(k^{(i)}, j)$  is the  $k^{th}$  user’s load of cluster “ $i$ ” at time “ $j$ ”.

Plot  $load(i), i \in [1, C]$  in a figure, load characteristic curve of different cluster marked with different colors. Figure 3 shows the load characteristic curve of each cluster by clustering the data of one day.

## 3.2 “Regression” stage

### 3.2.1 Constructing training datasets

In this section, we integrate the load characteristic curve of each cluster of  $(m-6)th, (m-7)th, (m-8)th, (m-14)th$  as the “attributes” and the real-time load as a “tag” of the predictable

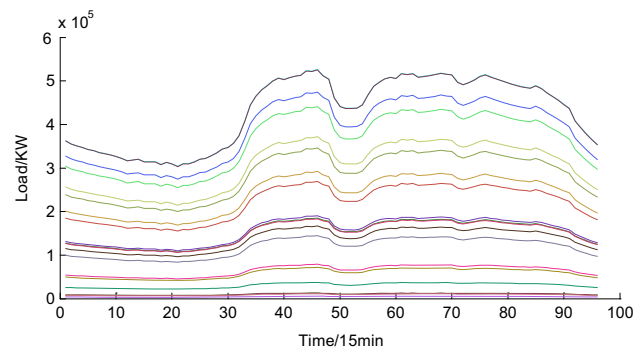


Fig. 3 Load characteristic curve of each cluster (by Apache Spark k-means algorithms)

day’s load. Notably, “regression” comes from “classification”, and the data label of “regression” is continuous, while the data label of “classification” is discrete. In order to test the validity and robustness of the “clustering–regression” model, we select the Spark decision tree and the Spark random forest as the comparison in the regression stage.

### 3.2.2 Regression prediction

The random forest algorithm is an ensemble algorithm, which is comprised of multiple single decision tree whose details are shown in Algorithm 3. Due to the diversity and unstable of results from individual decision tree, ensemble scheme is not easy to cause over-fitting and has the better prediction and classification performance.

In the Apache spark MLlib, the decision tree is wrapped in a Random Forest [32]. The regression procedures are explained as the follows:

- First establish a Random Forest containing one decision tree, and then grow the decision tree in the Random Forest. The method *train* is defined in object *DecisionTree* to set the parameters of the training model. The class *DecisionTree* is used for training classification and regression model, in the process of establishing the tree call the *run* method in class *RandomForest*, set the parameter *seed=0*, that only generate one decision tree.
- The method *buildMetadata* of object *DecisionTreeMetadata* calculate the number of possible splits and the number of leaf nodes that are generated. The method *findSplitsBins* of object *DecisionTree* search for each possible splitting attribute and the number of leaf nodes generated. Furthermore, the method *findBestSplit* of object *DecisionTree* finds the best splitting attribute, mainly through the method *binsToBestSplit* to determine the best splitting attribute.
- In the process of establishing a random forest, all of the trees need to be training by the method *run*. The deci-



sion tree training process is actually the decision tree construction process. The top-down recursive structure is used until meeting certain stop conditions.

Note that class *DecisionTreeModel* defined the method *predict*, but need to call the method *predict* of class *Node* to achieve the forecast.

Algorithm 3 DecisionTree(D, F)

Algorithm 3	DecisionTree(D, F)
1	Creat node $N$ ;
2	<i>loop</i> :
3	if {elements inset $D$ belong to same type} $\Rightarrow$ true then;
4	$N \rightarrow$ Leaf, labeled as a type of elements in $D$ ;
5	elseif {Attribute set $F=\emptyset$ } $\Rightarrow$ true then;
6	$N \rightarrow$ Leaf, labeled as the majority of the parent node Type;
7	else
8	<i>find_best_split</i> ();
9	the number of attributes in the attribute set $F$ reduced by 1;
10	for $\{i=1 \cdots n\}$ do
11	Create node $N_i$ for attribute value $a_i$ ;
12	if {Data set $D_i$ corresponding to $N_i=\emptyset$ } $\Rightarrow$ true, then
13	$N_i \rightarrow$ Leaf, labeled as the majority of the parent node type;
14	else
15	<i>goto loop</i>
16	endif
17	endfor
18	endif

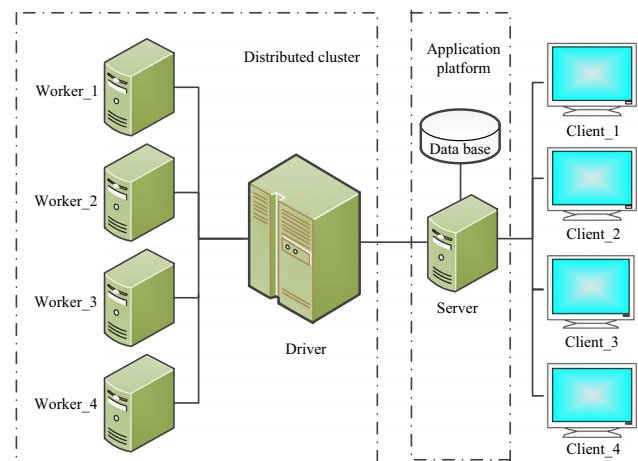


Fig. 4 Topology of distributed cluster

## 4 Experimental results

### 4.1 Distributed cluster construction based on Apache Spark platform

We use a distributed platform constructed in the laboratory to give the comprehensive simulation experiments. The distributed cluster use a DELL server, with 12 core CPU, clocked at 2.6 GHZ, 32 G memory, 10 T hard disk, Linux Ubuntu12.04 desktop operating system, as a management node driver. 4 Lenovo ordinary desktop, with 2 core CPU, clocked at 2.94 GHZ, 2 G memory, 1T hard disk, Linux Ubuntu12.04 desktop operating system, is used as a computing node worker. The open source software versions of the cluster include Hadoop version 2.7.3, Spark version 2.0.1, jdk version 1.8, Scala version 2.11.8 and python version 2.7. The complete experimental framework is shown in Fig. 4.

### 4.2 Experimental design and analysis

The “user grouping raw data” has been introduced in Sect. 3.1. The “electrical power load data” is automatically collected the load values of the users in the region at intervals of 15 min by the metering automation system, producing 96 records one day.

According to **Algorithm 1**, the process of the “cluster-regression” model is performed. The details of the steps have been shown in Sect. 3. The error evaluation of model regression prediction can be calculated by the mean absolute percentage error (MAPE) and root-mean-square error (RMSE).

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{x_i - \hat{x}_i}{x_i} \right| \times 100\% \quad (9)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n \left( \frac{x_i - \hat{x}_i}{x_i} \right)^2} \times 100\% \quad (10)$$

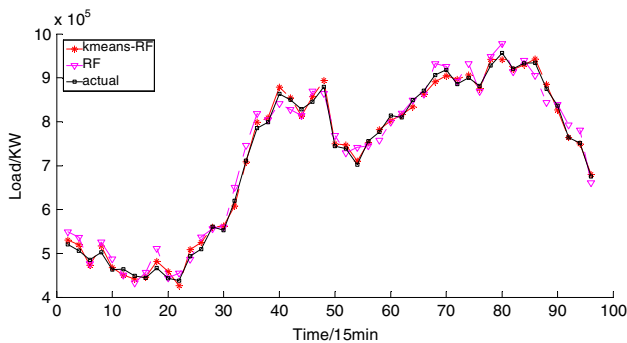
where  $x_i$  is the actual value of the time  $i$ -th load, and  $\hat{x}_i$  is the corresponding predicted value.

In order to test the validity and robustness of the “clustering–regression” model, the comprehensive experiments include the following four cases.

**Experiment 1:** Examining the validity of the “clustering–regression” model.

In our experiments, we compare proposed scheme with the scheme that use random forest regression directly. The data from a certain substation 2016-03-01 to 2016-08-25 are used as the training set to predict the load of a line of 2016-08-26, and the results are shown in Fig. 5.

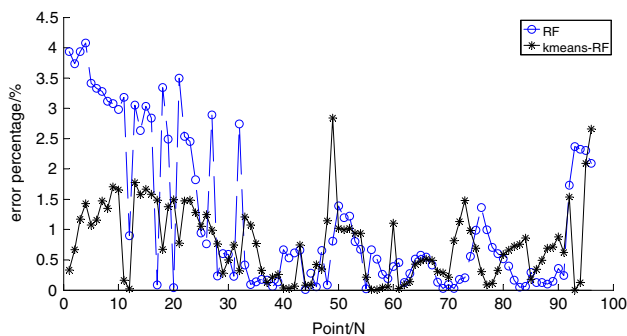
With Eqs. (9) and (10), the error of two load forecasting methods is calculated and the corresponding MAPE and RMSE on the test set are shown in Table 2. Moreover, the predicted percentage error curve for each algorithm at each metering point are also shown in Fig. 6.



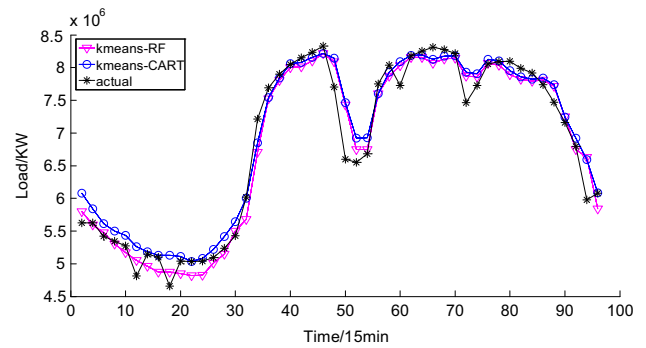
**Fig. 5** Comparison of load forecasting results between “clustering–regression” (k-means-RF) model and RF model

**Table 2** Comparison of load forecasting accuracy “clustering–regression” (k-means-RF) model and RF model

Evaluation index	Direct regression RF (%)	“clustering–regression” kmeans-RF(%)
MAPE (%)	3.163	2.040
RMSE (%)	4.473	3.145



**Fig. 6** Comparison of load forecasting error percent between “clustering–regression” (k-means-RF) model and RF model



**Fig. 7** Comparison of load forecasting results that with different combination of “clustering–regression” model

As can be seen from Table 2, Figs. 5 and 6, the load forecasting method using the “cluster-regression” model is superior to the general regression prediction model in the test set. Two reasons can explain this phenomenon: (1) Proposed scheme uses the “clustering–regression” model of k-means-RandomForest (k-means-RF). Although the RF-based load forecasting method shows excellent learning ability and adaptability [33–35], and has a relatively simple structure and lower spatial structure complexity, proposed scheme clustering the users by electrical features and got the “load characteristic curve of each cluster” and considered as factors in “regression” stage. As can be seen from Figs. 6 and 7, the prediction accuracy of “clustering–regression” model is higher than general regression prediction model. (2) The “clustering–regression” model proposed in this paper does not directly uses the historical load data, but analyzes the composition of the load. Since different types of users have different electrical power characteristics, the contribution for the total load is also different. This fine-grained analysis leads to an improvement of prediction.

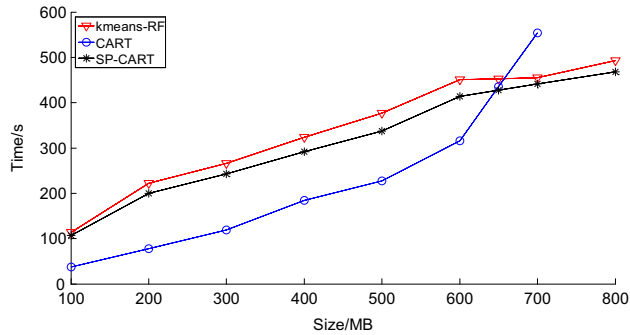
**Experiment 2:** Robustness Analysis of “clustering–regression” Model.

The “clustering” stage uses k-means algorithm, while the “regression” stage use two algorithms: (1) classification and regression tree (CART) and (2) the FR, so the “clustering–regression” model produces two predictive datasets. The data from a certain substation 2016-03-01 to 2016-08-25 are used as the training set to predict one line’s load of 2016-08-26, and the results are shown in Fig. 7. Also, the error statistics are shown in Table 3.

With the analysis of the results in Table 3 and Fig. 7, we can obtain two conclusions: (1) It can be seen that the forecasting result of “clustering–regression” model is reasonable, according to the two predictive datasets. The model is rather robust. (2) k-means-RF model based on RF has higher prediction accuracy than that of k-means-CART in the “regression” stage. This is because the RF combines the “Bagging” method to make decision tree together, the

**Table 3** Experiment's result of "clustering–regression" model

Clustering	Regression			
	Classification and regression tree CART		Random forest	
	MAPE	RMSE	MAPE	RMSE
K-means	4.531	5.776	2.243	3.350

**Fig. 8** Comparison of running time between algorithms based on distributed cluster and based on single machine

establishment of the model not only retain the statistical characteristics of the original data set, and in the establishment of the model as much as possible to reflect the randomness of attributes set. RF is more robust to errors and outliers. With the increasing the number of trees in the forest, the generalization error of random forest is converged, and the risk of over-fitting is reduced. From Fig. 8, the load curve predicted by the k-means-RF model is closest to the actual load.

**Experiment 3:** Testing and verifying the “clustering–regression” model with a larger data set. The size of experimental data set is shown in Table 4. Since the experimental data is limited and original data will be artificially expanded, different sizes of data files are used into our experiments.

The five data sets are incremented in succession, k-means-RF is run on each data set, and MAPE is calculated according to Eq. (5). Table 5 shows the results. We can see that with the increasing of the data volume, the prediction error increases slightly. Furthermore, the maximum MAPE is 1.81% and the minimum MAPE is 1.65%. This demonstrates that the

**Table 4** Data sets with different volume

Data set	1	2	3	4	5
File size/GB	0.5	1.0	2.0	4.0	8.0
Items	$5 \times 10^6$	$1.0 \times 10^7$	$2.0 \times 10^7$	$4.0 \times 10^7$	$8.1 \times 10^7$

**Table 5** Load forecasting MAPE in different data sets

Items	$5 \times 10^6$	$1.0 \times 10^7$	$2.0 \times 10^7$	$4.0 \times 10^7$	$8.1 \times 10^7$
MAPE/%	1.77	1.65	1.78	1.74	1.81

fluctuation of forecasting error is smoothly and irregularly. As results, experiments show that proposed model can be applied in big data platform on load forecasting.

**Experiment 4:** Constructing the distributed cluster to solve the problem that the stand-alone model cannot be processed in the face of massive data.

In order to test this feature of the distributed computing platform, a set of stress tests was designed. Compared with the single-alone model, the proposed model in the distributed cluster is tested. The model in the distributed cluster uses the “clustering–regression” model (k-means-RF) and the spark classification and regression tree (SP-CART), while the single-alone model uses the CART. The establishment time of model in the three algorithms on the same data set is examined. The worker nodes running the three algorithms share the same CPU clock and the same operating system (Ubuntu12.04). The worker node running the single-alone CART, has 4 G memory. The two worker nodes running the k-means-RF and the SP-CART have 2 G memory. So the distributed cluster memory is totally 4 G either. Three algorithms use the same training data sets, whose size is 100 MB, 200 MB . . . up to 800 MB. The testing results are shown in Table 6 and Fig. 8. Single-alone CART model has an inefficient memory error when analyzing 800MB datasets. This demonstrates that single-alone CART model is not suitable for analyzing bigger data sets (Table 6).

From Table 6 and Fig. 8, we can also see that the load forecasting method based on distributed cluster has no speed advantage when the data set is small (less than 650 M). The stand-alone CART load forecasting method has a shorter run time than that of distributed cluster k-means-RF and SP-CART methods when the data set is less than 650 M. This is because the distributed cluster task allocation and scheduling needs a certain time. The single-alone model is restricted by machine memory, and it is grandly saturation up to 700



**Table 6** Running time of algorithms based on distributed cluster and based on single machine

File size/MB	Running time (T/s)		
	Single-alone	Distributed cluster	
	CART	SP-CART	k-means-RF
100	37	107	114
200	78	200	222
300	119	243	266
400	184	292	324
500	227	337	377
600	316	414	451
650	435	428	453
700	554	441	455
800	–	468	493

M. When data is up to 650 M, both the single-alone model and the distributed cluster model have the approximate running time. When data is more than 700 M, the single-alone model memory is insufficient and cannot be used. Moreover, with the increasing of data set, the computational time of distributed clustering based on distributed cluster basically increases linearly. As long as the distributed cluster size is expanding, the processing capacity of the cluster can ignore the affected by the size of the data set.

In addition, comparing the time curves of k-means-RF model with SP-CART model in Fig. 8, it can be found that the time curves of k-means-RF is slightly higher than SP-CART, but the height difference is substantially stable. This phenomenon can be explained as follows. On one hand, the “clustering–regression” model needs to “clustering” before “regression”, while the SP-CART model only needs “regression”. On the other hand, at the “regression” stage, the RF algorithm needs to “bagging”, while the SP-CART algorithm does not need. Compared to the algorithm running time, this part of the time consuming can be ignored. As the amount of computation nodes increases, the algorithm running time will be reduced. In summary, the current distributed clusters still have a great performance improvement.

## 5 Conclusion and future work

Based on the analysis of users’ electrical power characteristics, a clustering–regression model of short-term load forecasting is proposed. By clustering users’ electrical power characteristics, we obtain the load characteristic curve of each cluster. Relating to the forecast day, the load characteristic curve of each cluster is collected and the regression is made. The Apache Spark MLlib is used to realize proposed “clustering–regression” model.

With the actual load data of a power company in China, the comprehensive experiments are designed. The accuracy of load forecasting of proposed model is significantly improved. For the model, the combinations of different algorithms are used in the “clustering” and “regression” stages, the validity is verified. Since the model is to be used in distributed cluster, it can be tested with a large-capacity dataset, and is demonstrated to be robust. Compared with single-alone model for stress testing, the proposed model has obvious advantage in the background of big data and provides significant engineering practice. Although the proposed model has some advantages, the structure of the “clustering–regression” model can be further optimized. Further optimization will be considered to improve the data processing capacity of the “clustering–regression” model and make it suitable for medium or long term power load forecasting.

**Acknowledgements** This work was supported by National Natural Science Foundation of China under Grants (Nos. 61472236, 61672337, 61602295, and 61562020), Natural Science Foundation of Shanghai (No. 16ZR1413100), and the Excellent University Young Teachers Training Program of Shanghai Municipal Education Commission (No. ZZsd115105).

## References

1. Cai, Y., et al.: Modeling and impact analysis of interdependent characteristics on cascading failures in smart grids. *Int. J. Electr. Power Energy Syst.* **89**, 106–114 (2017)
2. Verma, V., Kumar, A.: Cascaded multilevel active rectifier fed three-phase smart pump load on single-phase rural feeder. *IEEE Trans. Power Electr.* **32**(7), 5398–5410 (2017)
3. ZhenYa, L.: *Global Energy Internet*. China Electric Power Press, Beijing (2015)
4. ZhenYa, L.: *Technology of Smart Grid*. China Electric Power Press, Beijing (2010)
5. Song, D., Liu, X.: Medium and long-term electric power planning load forecasting based on variable weights gray model. In: Huang, B., Yao, Y. (eds.) *Proceedings of the 5th International Conference on Electrical Engineering and Automatic Control*, pp. 137–144 (2016)
6. Hassan, S., et al.: A systematic design of interval type-2 fuzzy logic system using extreme learning machine for electricity load demand forecasting. *Int. J. Electr. Power Energy Syst.* **82**, 1–10 (2016)
7. Soudari, M., et al.: Learning based personalized energy management systems for residential buildings. *Energy Build.* **127**, 953–968 (2016)
8. Lei, S.L., Sun, C.X., Zhou, X.X.: The research of local linear model of short term electrical load on multivariate time series. *Proceedings of the CSEE* **26**(2), 5 (2006)
9. Hu, R., et al.: A short-term power load forecasting model based on the generalized regression neural network with decreasing step fruit fly optimization algorithm. *Neurocomputing* **221**, 24–31 (2017)
10. Khwaja, A.S., et al.: Boosted neural networks for improved short-term electric load forecasting. *Electr. Power Syst. Res.* **143**, 431–437 (2017)
11. Liang, Y., et al.: Short-term load forecasting based on wavelet transform and least squares support vector machine optimized by improved cuckoo search. *Energies* **9**(12), 827 (2016)

12. Dudek, G.: Short-term load forecasting using random forests. In: Filev, D., et al. (eds.) *Intelligent Systems. Architectures, Systems Applications*, pp. 821–828. Springer, Cham (2015)
13. Lee, C.-W., Lin, B.-Y.: Application of hybrid quantum Tabu search with support vector regression (SVR) for load forecasting. *Energies* **9**(11), 873 (2016)
14. Spark, A.: <http://spark.apache.org> (2017)
15. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. *HotCloud* **10**, 95 (2010)
16. Lin, F.: *Research and Implementation of Memory Optimization Based on Parallel Computing Engine Spark*. Tsinghua University, Beijing (2013)
17. Rodrigues, L.M., et al.: Parallel and distributed Kmeans to identify the translation initiation site of proteins. In: *Proceedings 2012 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1639–1645 (2012)
18. Yang, Y.: An improved cop-kmeans clustering for solving constraint violation based on mapreduce framework. *Fundam. Inf.* **126**(4), 301–318 (2013)
19. Pandagale, A.A., Surve, A.R.: IEEE: Hadoop-HBase for finding association rules using Apriori MapReduce algorithm. In: *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (Rteict)*, pp. 795–798 (2016)
20. Lee, K.C., 채수성, and 김인아, Development of detection system of vocal tic symptoms using SVM algorithm in Spark. *Database Res.* **32**(3), pp. 115–127 (2016)
21. Wang, B., Wang, D., Zhang, S.: Distributed short-term load forecasting algorithm based on Spark and IPPSO-LSSVM. *Electr. Power Autom. Equip.* **36**(1), 117–122 (2016)
22. Ma Tiannan, N.X., Huang, Y.: Short-term load forecasting for distributed energy system based on Spark platform and multi-variable L2-boosting regression model. *Power Syst. Technol.* **40**(6), 8 (2016)
23. Xie, M., Ji, D.J.L.X.: Cooling load forecasting method based on support vector machine optimized with entropy and variable accuracy roughness set. *Power Syst. Technol.* **41**(1), 5 (2017)
24. Yaslan, Y., Bican, B.: Empirical mode decomposition based denoising method with support vector regression for time series prediction: a case study for electricity load forecasting. *Measurement* **103**, 52–61 (2017)
25. Meng, X., et al.: MLlib: machine learning in Apache Spark. *J. Mach. Learn. Res.* **17**, 1235–1241 (2016)
26. Siegal, D., et al., Smart-MLlib: a high-performance machine-learning library. In *2016 IEEE International Conference on Cluster Computing*, pp. 336–345 (2016)
27. Zhang, F., et al.: A distributed frequent itemset mining algorithm using Spark for Big Data analytics. *Iust. Comput. J. Netw. Softw. Tools Appl.* **18**(4), 1493–1501 (2015)
28. Zaharia, M.: Apache Spark: a unified engine for big data processing. *Commun. ACM* **59**(11), 56–65 (2016)
29. Sepasi, S.: Very short term load forecasting of a distribution system with high PV penetration. *Renew. Energy* **106**, 142–148 (2017)
30. Zhang, S., Liu, J., Zhao, B., et al.: Cloud computing-based analysis on residential electricity consumption behavior. *Power Syst. Technol.* **37**(6), 1542–1546 (2013)
31. Tan, P.N., Steinbach, M., Kumar, V.: *Introduction to Data Mining*. Pearson Education, New York (2006)
32. Huang, M.: *Spark MLlib Machine Learning: Algorithm, Source Code and Practical*. Publishing House of Electronics Industry, Beijing (2016)
33. Gonzalez, C., Mira-McWilliams, J., Juarez, I.: Important variable assessment and electricity price forecasting based on regression tree models: classification and regression trees, Bagging and Random Forests. *LET Gener. Transm. Distrib.* **9**(11), 1120–1128 (2015)
34. Huang, N., Lu, D., Xu, D.: A permutation importance-based feature selection method for short-term electricity load forecasting using random forest. *Energies* **9**(10), 767 (2016)
35. Lahouar, A., Slama, J.B.H.: Day-ahead load forecast using random forest and expert input selection. *Energy Convers. Manag.* **103**, 1040–1051 (2015)



of the Machine Learning Technical Committee of the Chinese Association of Artificial Intelligence.



**Jingsheng Lei** received the B.S. degree in mathematics from Shanxi Normal University, in 1987, and the M.S. and Ph.D. degrees in computer science from Xinjiang University, in 2000 and 2003, respectively. His research interests include machine learning, data mining, pattern recognition, and cloud computing. He is a member of the Artificial Intelligence and Pattern Recognition Technical Committee of the China Computer Federation, and a member of the Machine Learning Technical Committee of the Chinese Association of Artificial Intelligence.

**Ting Jin** was born in 1982. She is an Associate Professor at the College of Information Science & Technology, Hainan University. She received her Ph.D. in computer software and theory from Fudan University. Her research interests include data mining, information retrieval and machine learning.



**Jiawei Hao** received the B.E. degree in electrical engineering from Taiyuan University of Technology in 2014. He is currently a postgraduate with the College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China. His current research interests include load forecasting, power big data, machine learning and data mining.



**Fengyong Li** received the M.S. degree in School of Information and Engineering in 2010 from Zhengzhou University, and the Ph.D. degree in School of Communication and Information Engineering in 2014 from Shanghai University. He is currently a lecturer of the College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China. His research interests include multimedia security, machine learning.