Low Voltage Prediction Based on Spark and Ftrl

Chen Gao[1, a], Zhongan Ding[1], Shengteng Yan[1] and Hongkun Mai [2]

[1]Power Science Research Institute of Fujian Electric Power Co., Ltd., Fuzhou 300007, China;

[2]China Satellite Marine Tracking and Control Department, Jiangyin 214400, China.

[a]sgaochen08@139.com

Keywords: Ftrl, Low voltage, Distributed, Big data.

Abstract. "Big data" is a popular keyword in recent years, and data related to the power grid mainly character huge amount, high complexity and broad sources. As low voltage has great influence on normal daily power utilization, besides good real-time monitoring and exception handling, a model based on big-data algorithms and multi-dimensional characters is also needed for real-time prediction. This paper advances a FTRL algorithm based on the Spark framework, and establishes a highly fault-tolerant, real-time, accurate and fast low-voltage prediction system by setting up a FTRL real-time computation platform based on effective characters extracted from a huge amount of voltage data. It can be seen through analysis of experimental results that this system is able to effectively predict low voltage and give alarms, which shows great improvement over the current manual monitoring mechanism.

1. Introduction

With increasingly developed industrialization, electrical equipment has become an essential element in daily life. Power grid systems are developed with increasing complexity, and improved gradually. Frequent electrical loads lead to low voltage of different levels in various power distribution regions. Such low voltage can not only influence effective running of normal equipment, but also cause malfunction or even interruption of industrial machines (converters are more sensitive to voltage).

Low voltage in power distribution networks can be caused by multi-dimensional factors. Possible causes are concluded as follows:

(1) Seasonal factors: as in many regions, electrical loads have wide ranges of change influenced by weather and temperature, power grid systems, power distribution systems are expected to intelligently predict and adjust voltage based on historical data to regulate the voltage within a reasonable range.

(2) Human factors: such factors are more noticeable in rural villages and towns. Especially, in some rural areas, low-voltage lines are of the three-phase four-wire type, and electricity is commonly connected through wires of utility poles by construction workers, resulting in imbalanced three phases.

(3)Geographical factors: different regions, such as industrially advanced cities, green cities, schools, factories and companies, have different modes of power utilization, resulting in different peak hours for electricity.

Such factors, together with massive historical data, can be analyzed to extract effective characters related to low voltage. Such characters processed through the big-data framework and the algorithmically optimized model can provide effective supports for the prediction of low voltage.

2. Background

The concept of big data has been integrated into various industries, and currently it has been widely used in power grid enterprises. Historical data related to low voltage are also massive, and therefore distributed storage systems, computing frameworks and high fault-tolerance mechanisms are needed to guarantee accurate, real-time and safe prediction and warning of low voltage.

At present, some relatively mature researches have been done in the industry, regarding low voltage prediction based on big data. For example, low voltage is predicted through logistic regression and a self-organizing neural network, and voltage is monitored in real time through a data analysis platform established based on data processing of an electricity information system. However, traditional batch processing algorithms fail to effectively handle super-large-scale power grid data and online data flow. In response to this problem, this paper puts forward a FTRL algorithm based on Spark for low voltage prediction [1-2].

3. Research Content

3.1 Spark Mechanism.

Spark adopts the classical Master-Slaver mode of the distributed framework. Master works as the main control unit of a cluster, to schedule and execute all tasks of the whole cluster. Worker refers to a computational node, which executes scheduling instructions from Master and reports status to the main control node through heartbeat on a regular basis. Executor is in charge of execution of task scheduling. Client as the user end is to submit an application, and Driver takes charge of the execution of an application [3]. The structure chart is shown in figure 1:

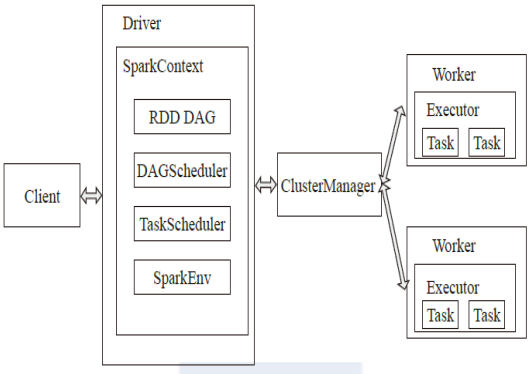

Fig. 1 The framework of spark

3.2 Ftrl.

With regard to large-scale power distribution data, traditional offline batch processing methods fail to satisfy requirements on processing performance and time, and especially in the scenario of low-voltage prediction, proper online processing methods are expected to solve the problem. The development of Ftrl is good to the online optimization and it can effectively solve the problem of sparseness commonly appeared in high dimensional data [4].

The advantage of online learning in low-voltage prediction is that as for each new voltage character sample, its loss and gradient produced can be used to iterate the existing incremental model, and one-by-one real-time training can satisfy scenarios with large-scale data. Figure 2 shows the realization of FTRL in engineering context. Every dimension of W is trained and updated separately with a different learning rate, which is indicated by the item before lamda2 in codes. Compared with using the same learning rate for all characters of W, this method can fully solve uneven distribution of sample data among characters with different types. For example, during low-voltage prediction, training samples regarding characters in the weather dimension are less, and thus relevant training rate can be set with a bigger value, so that as for a new training sample, this dimension will have a bigger gradient and doesn't need to maintain a unified pace with other dimensions. It is very suitable for scenarios with uneven distribution of samples [5]. Pseudo-codes for engineering realization of FTRL are shown in figure 3.

Algorithm 1 *Per-Cooedinate FTRL-Proximal with L_1 and L_2 L_2 Regularrization for Logistic Regression*

 # with per-coordinate learning Regression

Input: parameters $\alpha, \beta, \lambda_1, \lambda_2$

$(\forall i \in \{1,\ldots,d\})$, *initialize* $z_i = 0$ *and* $n_i = 0$

for $t = 1$ *to* T *do*

Receive feature vector x_t *and let* $I = \{i \mid x_i != 0\}$

for $i \in I$ *compute*

$$w_{t,i} = \begin{cases} 0 & \text{if } |z_i| <= \lambda_1 \\ -(\frac{\beta + \sqrt{n_i}}{\partial} + \lambda_2)^{-1}(z_i - \text{sgn}(z_i)\lambda_1) & \text{otherwise} \end{cases}$$

Predict $p_t = \sigma(x_t.w)$ *using the* $w_{t,i}$ *computed above*

for all $i \in I$ *do*

$g_i = (p_t - y_t)x_i$ *# gradient of loss w.r.t* w_i

$\sigma_i = \frac{1}{\sigma}(\sqrt{n_i + g_i^2} - \sqrt{n_i})$ *# equals* $\frac{1}{\eta_{t,i}} - \frac{1}{\eta_{t-1,i}}$

$z_i <- z_i + g_i - \sigma_i w_{t,i}$

$n_i <- n_i + g_i^2$

end for

end for

Fig. 2 The pseudocode of ftrl

3.3 Set Up Characters.

Hierarchical classification is conducted towards various influence factors of voltage by synthesizing original data and based on practical scenario conditions. Character engineering is then established. Effectiveness of characters can be verified through offline data, and data with good evaluation effect will be added to the online real-time model [6]. The dimensionality of character is shown in figure 3.

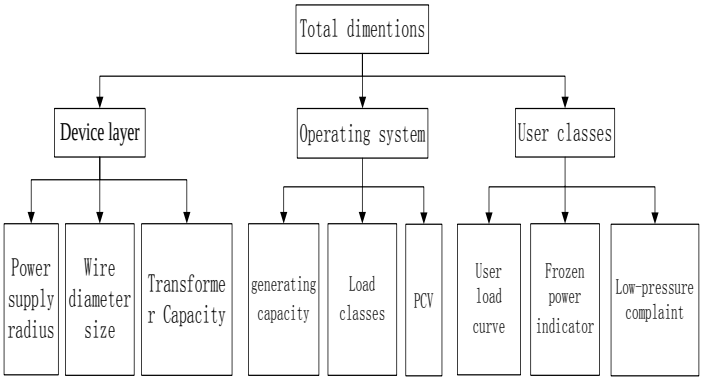

Fig. 3 The dimensionality of character

3.4 Spark Processing Data.

As for massive power-grid data, the first thing to be considered is the mechanism for data storage. Traditional relational databases can no longer meet the storage requirements. Hive is a data warehouse based on Hadoop, with the same query functions as ordinary SQL. Query statements of ordinary SQL can be executed in distributed clusters after they are converted into Map-Reduce tasks. Data are stored in terms of date partition. Hive tables are created as per relevant rules to correspond to data of different types. Real-time data of a day will be written into the partition corresponding to the date of this day. In this way, data can be loaded in terms of partitions at later stages, realizing obviously faster data accessing of Spark.

The character platform conducts Spark task scheduling with regard to power-grid data on a daily basis. Firstly various dirty data need to be eliminated, such as lost voltage, current and collection-point data. A Map task should be started to filter out missing data, before Spark reads data of Hive partitions, to guarantee completeness of samples. Normalization should be applied during character extraction to speed up convergence rate of the algorithm. In this way, data will be exported to relevant Hive tables in a standardized manner, and also stored according to partitions.

During character extraction, there is a factor causing big influence on performance in Spark that must be considered, which is the data skew. Figure 4 shows data skew in Spark tasks. Data produced

in different systems of the power grid are stored in different Hive tables, and Join operations should be conducted towards such data during character extraction. However, sometimes, most tasks are executed fast while some others are executed slowly, or an error of OOM (out of memory) is reported suddenly during normal execution of Spark jobs.

85	154	0	SUCCESS	PROCESS_LOCAL	3 / rz-data-hdp-dn0912.rz.sankuai.com	2016/01/29 13:42:02	3.2 min	0.9 s	807.7 KB / 8691
86	155	0	SUCCESS	PROCESS_LOCAL	46 / rz-data-hdp-dn0890.rz.sankuai.com	2016/01/29 13:42:02	49 s	0.5 s	531.4 KB / 5309
87	156	0	SUCCESS	PROCESS_LOCAL	92 / rz-data-hdp-dn1275.rz.sankuai.com	2016/01/29 13:42:02	31 s	0.6 s	360.7 KB / 3696
88	157	0	SUCCESS	PROCESS_LOCAL	64 / rz-data-hdp-dn0121.rz.sankuai.com	2016/01/29 13:42:02	27 s	0.4 s	406.1 KB / 4104
89	158	0	SUCCESS	PROCESS_LOCAL	13 / rz-data-hdp-dn1184.rz.sankuai.com	2016/01/29 13:42:02	14 s	0.4 s	347.3 KB / 3561
90	159	0	SUCCESS	PROCESS_LOCAL	5 / rz-data-hdp-dn0912.rz.sankuai.com	2016/01/29 13:42:02	13 s	0.3 s	351.4 KB / 3622
91	160	0	RUNNING	PROCESS_LOCAL	90 / rz-data-hdp-dn0059.rz.sankuai.com	2016/01/29 13:42:02	3.9 min	0.8 s	1617.0 KB / 18545
92	161	0	SUCCESS	PROCESS_LOCAL	87 / rz-data-hdp-dn0879.rz.sankuai.com	2016/01/29 13:42:02	26 s	0.4 s	318.1 KB / 3081
93	162	0	SUCCESS	PROCESS_LOCAL	55 / rz-data-hdp-dn0875.rz.sankuai.com	2016/01/29 13:42:02	19 s	0.5 s	359.6 KB / 3574
94	163	0	RUNNING	PROCESS_LOCAL	82 / rz-data-hdp-dn0430.rz.sankuai.com	2016/01/29 13:42:02	3.9 min	0.8 s	2023.4 KB / 22812
95	164	0	SUCCESS	PROCESS_LOCAL	99 / rz-data-hdp-dn0817.rz.sankuai.com	2016/01/29 13:42:02	5 s	0.2 s	188.1 KB / 1426
96	165	0	SUCCESS	PROCESS_LOCAL	56 / rz-data-hdp-dn0875.rz.sankuai.com	2016/01/29 13:42:02	10 s	0.3 s	214.5 KB / 1683
97	166	0	SUCCESS	PROCESS_LOCAL	71 / rz-data-hdp-dn0576.rz.sankuai.com	2016/01/29 13:42:02	2.9 min	0.4 s	673.8 KB / 6932
98	167	0	SUCCESS	PROCESS_LOCAL	77 / rz-data-hdp-dn0242.rz.sankuai.com	2016/01/29 13:42:02	13 s	0.3 s	276.3 KB / 2349
99	168	0	RUNNING	PROCESS_LOCAL	58 / rz-data-hdp-dn0491.rz.sankuai.com	2016/01/29 13:42:02	3.9 min	1 s	1321.0 KB / 14508

Fig. 4 The model of low voltage prediction

The problem of data skew can be effectively solved through following methods: filter out minority Keys that have far more corresponding values than the standard; improve the parallelism of Shuffle operations; split Keys that cause data skew and then conduct relevant Join operations.

3.5 Page Numbers.

Establishment of the model includes two parts: offline training and online real-time computation, as shown in figure 5. Offline training is mainly used for fault-tolerant backup. Once faults occur during online real-time computation, training data will be written into the cache. However, if the cache is faulty or some data is abnormal, historically accumulated weight data will be lost, which can't be rolled back. Thanks to the offline training mechanism, previously accumulated training weight data can be imported online from offline, when an online task is faulty.

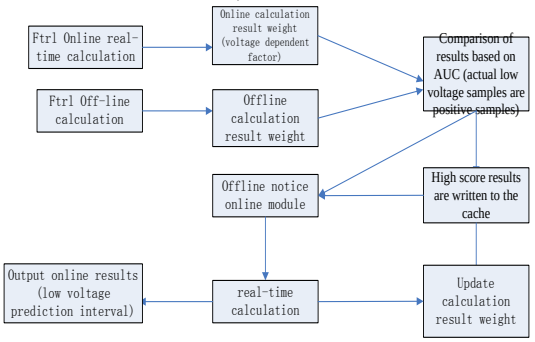

Fig. 5 The model of low voltage prediction

Offline training is carried out in an incremental manner and on a daily basis. Training data samples are established according to characteristic indexes that influence the voltage quality, which will then be completed by continuously adding typical data of transformer working areas. Data sets are divided into test sets and validation sets, all of which will be continuously optimized. Parameters with best AUC will be converted into Json strings and then saved in Hive, through Spark tasks. For new training samples every day, firstly historical training result in Hive will be read and Json analysis will be conducted for incremental training. Daily training results are stored in partition tables, and this is a good measure for fault-tolerant backup, as in this way poor training effect of a single day will not influence historical training results. Training results of any day in history can be picked up for incremental training, together with new samples [7].

The online FTRL algorithm fully considers distinctions of Fobos and RDA, regarding regularization terms and W [8]. After a new sample is processed, the weight is updated according to formula (1):

$$w^{(t+1)} = \arg\min_{W}\left\{G^{(1:t)}.W + \lambda_1 \| W \|_1 + \lambda_2 \frac{1}{2} \| W \|_2^2 + \frac{1}{2}\sum_{s=1}^{t}\sigma^{(s)} \| W - W^{(s)} |_2^2\right\} \quad (1)$$

In this formula, there is a L2 regularization term, $\frac{1}{2}\| w \|_2^2$. The introduction of this regularization term will not influence the sparseness of FTRL, and instead it will produce smoother results [9]. Although the formula looks complicated, it is actually equivalent to solving an optimization problem after it is adapted and expanded.

$$W^{(t+1)} = \arg\min_{W}\{(G^{(1:t)} - \sum_{s=1}^{t}\sigma^{(s)}W^{(s)}).W + \lambda_1 || W ||_1 + \frac{1}{2}(\lambda_2 + \sum_{s=1}^{t}\sigma^{(s)}) || W |_2^2 + \frac{1}{2}\sum_{s=1}^{t}\sigma^{(s)} || W^{(s)} |_2^2\} \quad (2)$$

With regard to various dimensions of character weight, split it into N independent questions of scalar minimization. The algorithm has a very fast iterative speed, which is very practical for online computation. With this algorithm, low-voltage prediction can process data and feedback the predicting results to the business end in real time [10].

Online FTRL can compute sample data in real time to obtain results related to low-voltage levels, and it can convert results with a regression model and based on historical data to predict.

4. Analysis

To verify performance and correctness of the model, several typical data are selected to conduct relevant training and testing. Figure 5 shows the format of data. Characters includes regarding power supply units, number of collection points, number of transformer work areas, and dates of original voltage data, among which data dates are used for storage in designated partitions of Hive. Data will be accessed according to partitions during offline incremental computation.

Table 1. Format List of Data

Character_1	Character_2	Character_n-1	Character_n
T11	T12	T1(n-1)	T1n
T21	T22	T2(n-1)	T2n
T(k-1)1	T(k-1)2	T(k-1) (n-1)	T(k-1)n
Tk1	Tk2	Tk(n-1)	Tkn

As there are many dirty data among original data, such as data missing, exception and outliers, a timed Spark task should be enabled for data pretreatment once data from different areas are collected every day. As to lost data, the smoothing method should be used by taking mean values, for data completion. Abnormal data should be eliminated to avoid negative influence on the model. Data finally generated should be normalized. On one hand, differences among values of character data can be taken into account, and on the other hand, convergent accuracy of the algorithm can be improved as well.

Figure 6 shows character weight of offline training results, which have been converted into Json strings and stored in partition tables. As offline training results of a day is obtained through Json analysis and incremental computation based on results of the previous day, relevant computing costs can be reduced. According to the derivation process of FTRL algorithm, resulting parameters can be divided into weight, independent learning rate for each character dimension, which can be reflected by the gradient component of a dimension, and two additional intermediate parameters that are used together during iteration.


```
"w":{"m":[{"k":"69","v":"0.534504"},{"k":"200409","v":"0.196539"},
{"k":"96","v":"0.100315"},{"k":"138","v":"0.0874326"},{"k":"132","v":"0.049071"},{"k":"120","v":"0
{"k":"271","v":"-0.00919417"},{"k":"2","v":"-0.0105255"},{"k":"7","v":"-0.0186738"},{"k":"203","v"
{"k":"205","v":"-0.0716393"},{"k":"63","v":"-0.0816593"},{"k":"1","v":"-0.0976296"},{"k":"299","v"
"z":{"m":[{"k":"109","v":"1.42185"},
{"k":"200606","v":"69.7288"},{"k":"200409","v":"-14.0165"},{"k":"200605","v":"-0.0479491"},{"k":"6
{"k":"120","v":"-0.820447"},{"k":"63","v":"1.37676"},{"k":"57","v":"0.869833"},{"k":"7","v":"0.701
{"k":"1","v":"2.27214"},{"k":"203","v":"0.739809"},{"k":"2","v":"0.661075"},{"k":"205","v":"1.4584
{"k":"41","v":"0.344665"},{"k":"206","v":"0.132942"},{"k":"45","v":"0.254102"},{"k":"9","v":"0.182
```


Fig. 6 Character Weight of Offline Training Results

Online FTRL algorithm can compute real-time sample data, update results, and covert results into Json strings to store them in Hive in real time, with a level of milliseconds. When online computation fails, results can be loaded through offline and then recovered to guarantee good fault tolerance [11].

Figure 7 shows the relationship between training time and AUC. It can be seen that data within a month have more obvious increase while those exceeding a month have slower increase, with a tendency to a stable value.

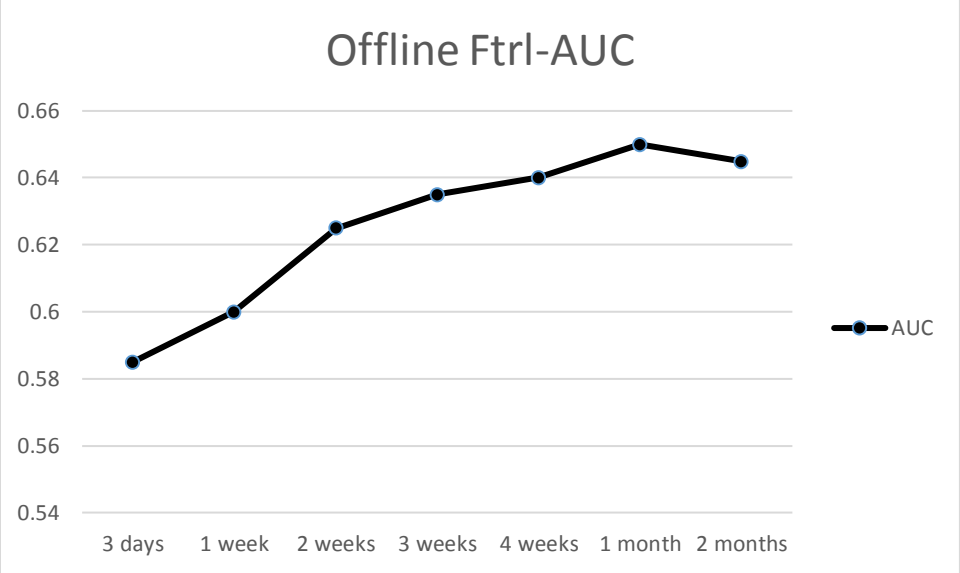

Fig. 7 The relationship between time and AUC

Effect of online computation is not significant at the earlier stage of data accumulation. The cold start problem can be well solved at the earlier stage by loading and overwriting offline data. When the algorithm is started, better effect will be shown as a result.

5. Summary

This paper mainly aims to establish a low-voltage prediction system based on the technology of big data. Power-grid data are normally massive, complicated and diversified, with frequently appeared abnormal data. Spark is selected as the main framework for establishment of the character extracting model, due to its fast in-memory computation and good fault tolerance. Data generated by different power-gird systems are stored in Hive partition tables, for extraction of effective characters. Offline training and online real-time computing are combined to guarantee high back-off and fault-tolerance of the system. The offline model is of incremental type and offline training is conducted on a daily basis to avoid resource shortage due to excessive data. When the online model for low-voltage prediction is faulty, data can be loaded from offline to guarantee data safety. From AUC for training with voltage data from some typical areas, it can be seen that AUC computed with the FTRL offline model is up to about65%, and that the FTRL online model has good timeliness, safety, reliability and stability, which is more suitable for low-voltage prediction, compared to batch training algorithms such as ordinary logistic regression and GBDT. Still, the current models have big space for optimization. For character engineering, a complete character system needs to be developed and then integrated into the current system for character mining, based on techniques related to deep learning. Spark streaming can be used in online FTRL computation to guarantee more reliable timeliness, improve predicting accuracy and achieve faster response. Next, relevant researches will be further conducted.

References

[1]. Leong F T L, Austin J T. The psychology research handbook: A guide for graduate students and research assistants [M]. SAGE publications, 2005.

[2]. Kekez, M. M., Barrault, M. R., & Craggs, J. D. (1972). Spectroscopic investigation of the spark channel. Journal of Physics D: Applied Physics, 5 (2), 253.

[3]. Barbara P, Cawthorne A B, Shitov S V, et al. Stimulated emission and amplification in Josephson junction arrays [J]. Physical review letters, 1999; 82 (9): 1963.

[4]. Miliotis P A. Data envelopment analysis applied to electricity distribution districts [J]. Journal of the Operational Research Society, 1992; 43 (5): 549-555.

[5]. Alahakoon D, Yu X. Advanced analytics for harnessing the power of smart meter big data [C]//Intelligent Energy Systems (IWIES), 2013 IEEE International Workshop on. IEEE, 2013: 40-45.

[6]. Stathis J H, DiMaria D J. Reliability projection for ultra-thin oxides at low voltage [C]//Electron Devices Meeting, 1998. IEDM'98. Technical Digest. International. IEEE, 1998: 167-170.

[7]. Yu Yong-jun, Qi Xiao-xiao, Nan Dong-liang, et.al. Application of Large Data in Low Voltage Evaluation of Distribution Network [J]. Electrical Technology, 2015; 10 (10): 92-94.

[8]. Zhang Peng-fei, Xu Zhi-yu, Xu Tang-yun, et.al. Analytical mining and computing paradigm for large data of smart grid [J]. Electrical Technology, 2015 (2): 89-94.

[9]. Huang Xiao-qing, Chen Jie, Chen Yong-xin, et.al. Charging Station Load Forecasting Method under Data Background [J]. Automation of Electric Power Systems, 2016; 12: 010.

[10]. Wang Qi-ying. Power distribution solutions and energy efficiency management in the data center--the view of Engineers and manufacturers partners--UPS power supply system data center for power supply and distribution system design errors in the era of large data [J]. Electrical application, 2015; 34(15): 12-13.

[11]. Bai Yang. Research on information aggregation of power equipment condition monitoring for large data [D]. University of Science and Technology of KunMing, 2016