Eng Int Syst (2019) 1: 19–26

Engineering
Intelligent Systems

Analysis of Web Data Mining Combining Software Capability Maturity Model

Xiang Li[1]* **and Zijia Zhang**[2]

[1] *School of Artificial Intelligence and Big Data, Chongqing College of Electronic Engineering; Chongqing 401331 China*
[2] *School of Automation, Nanjing university of information science and technology; Nanjing Jiangsu 210044 China*

Based on the real-time requirements of the Java Web big data mining model and the diversity characteristics of data samples, a Java Web big data mining model for the software capability maturity model (hereinafter referred to as SCMM for short) is put forward. The calculation process for the construction of the current model is analyzed, and the construction process is divided into two stages namely the rough adjustment stage and the fine-tuning stage, according to the size of change of model vector. It is found that most of samples in the fine-tuning stage have very little influence on the calculation results. Therefore, it is not necessary to calculate the gradient of such samples in the fine-tuning stage, while the calculation results constructed previously can be used directly, thereby reducing the amount of computation and improving the computational efficiency. The experimental results show that the proposed model can reduce the amount of computation in the model training phase by about 35% in the distributed cluster environment. In addition, the model accuracy obtained by the training is within the normal scope, and the real-time performance of the Java Web big data mining model can be effectively improved.

Keywords: Mining algorithm; Distributed System; Machine Learning; Software Capability Maturity Model; Optimization Model

1. INTRODUCTION

The mining of real-time data information from the big data to assist in decision making is of great significance. [1–2]. It is essentially based on the machine learning and other models, such as the logistic regression, support vector machine etc., to establish the models through gradient descent [3–6]. Different from the traditional data mining applications, big data mining requires that the correctness of calculation results should be within a certain expected range. In addition, it has a strong requirement for the rapidity of data processing and the real-time performance of results [7–9]. Hence, it is necessary to make use of clusters to carry out the model construction in parallel, thereby improving the speed of computation [10–14].

The machine learning Java Web big data mining models at present can mainly be divided into two categories. In the first category, the existing single-machine gradient descent optimization model is applied in parallel and operated on the general distributed platform, such as the optimization model based on the multi-core parallel environment and MapReduce framework [15]. In the second category, the Java Web big data mining model is designed in accordance with the Java Web big data mining models based on the parameter server framework such as Piccolo [16], Adam [17], Petuum [18], Parameter Server [19] and so on. The flexible consistency model [20] is applied to improve the efficiency of model execution, and the approximate gradient descent method based on the bounded delay is put forward. These models often have problems such as relatively large synchronous waiting cost, slow convergence rate, large number of construction times and so on. In the context of the Java Web big data mining model, it is necessary to improve the current optimization model in accordance with the requirements for the real-time performance and model accuracy.

In this paper, a Java Web big data mining model is put forward, which makes use of the difference in the contribution

*Corresponding author: Email: favouriteme@sina.com

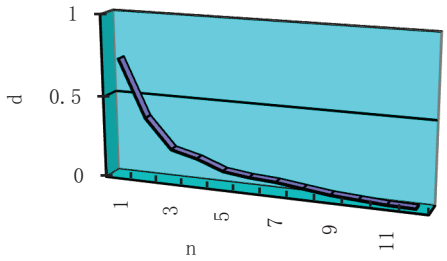

Figure 1 Amount of change in the model parameter vector during the construction process.

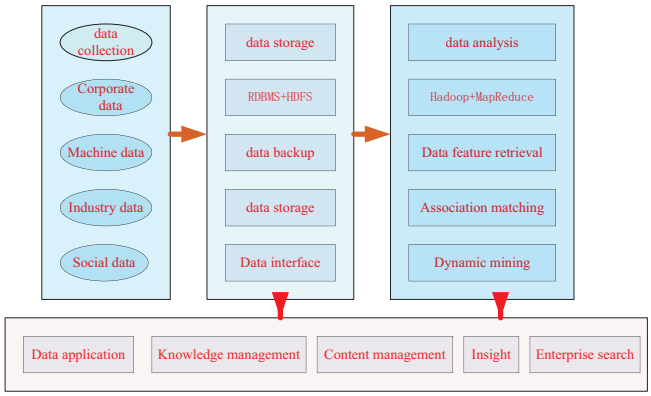

Figure 2 Architecture Diagram of Large Data Management Model.

of samples to the model update to achieve the reutilization of the intermediate results in the calculation, thereby reducing the computational complexity in the construction process, improving the computational efficiency of the Java Web big data mining model and the real-time performance of the results. To facilitate the description, the parameter server (PS) is taken as a research object to analyze the characteristics of the construction and calculation process and implement the construction model based on the software capability maturity model, which is further compared with the latest Java Web big data mining model.

2. CONSTRUCTION OF THE JAVA WEB BIG DATA MINING MODEL

2.1 Calculation Process for the Construction of the Software Capability Maturity Model

The calculation process of the software capability maturity model consists of construction calculation for multiple times, and the parameter vector of the model obtained through the update in the i-th construction is v_i, and the parameter vector of the model obtained in the $(i+1)$-th construction is v_{i+1}, then the angle between the vectors v_i and v_{i+1} is regarded as the amount of change in the parameter vector of the model in the $(i+1)$-th construction. The change in the parameter vector of the model in the whole construction process is shown in Figure 1. The horizontal coordinate indicates the number n of constructions, and the vertical coordinate indicates the amount d of change in the model vector.

From Figure 1, it can be known that in the initial stage of construction, the change in the parameter vector of the model is relatively large in each construction. However, in the later stage of construction, the difference of the model parameter vector between different constructs is relatively small. Hence, the threshold parameter can be set as α, when the construction results in the model parameter vector change greater than α, it is considered that the construction is in the rough adjustment stage, and the change in the model parameter is relatively large in the construction calculation each time. When the amount of change in the model parameter is smaller than α, the construction is in the fine-tuning stage. At this point, the construction is close to the optimal value, and the amount of change to the model parameter is relatively small in each construction.

Since in the fine-tuning stage, the update of the model parameter vector is relatively small in the construction calculation each time, the calculation process of the fine-tuning stage is mainly taken into consideration to reduce the amount of computation and improve the system computation efficiency and real-time performance.

This paper integrates data mining technology into large data processing and management, intelligently collects, analyses, manages and efficiently utilizes massive data with Hadoop and other open source technologies, and constructs a large data management model based on data mining. The model structure is shown in Figure 2.

In the calculation process, when the construction is performed each time, the corresponding gradient vector g_i is calculated based on each training sample x_i, and all the g_i are synthesized to obtain the total gradient g. However, in the synthesis of the gradient vector, the gradient vectors corresponding to different samples have different contributions to the total gradient.

Table 1 Distribution of the Contribution of Samples to the Model Update in Each Construction.

Number of iterations	Contribution interval of the samples to the model update					
	0∼0.1	0.1∼0.2	0.2∼0.3	0.3∼0.4	0.4∼0.5	0.5∼1.0
1	4.31	43.14	26.90	12.93	5.90	6.82
2	66.03	15.80	7.50	4.01	2.47	4.19
3	80.37	12.13	4.29	1.59	1.10	0.52
4	85.93	9.33	2.63	1.40	0.44	0.27
5	87.70	8.43	2.16	1.23	0.33	0.16
6	88.71	7.80	1.90	1.17	0.29	0.13
7	89.77	6.91	2.03	0.97	0.20	0.11
8	90.26	6.47	2.10	0.86	0.21	0.10
9	90.56	6.29	2.10	0.76	0.20	0.09
10	90.57	6.24	2.13	0.79	0.17	0.09
11	90.67	6.16	2.14	0.77	0.16	0.09
12	90.73	6.14	2.09	0.80	0.14	0.10

Table 2 Quantity Distribution of Two Types of Samples in the Fine-Tuning Stage and Their Contribution to the Model Update.

Number of iterations	Sample proportion/%		Contribution to the model update	
	$\leqslant \beta$	$\rangle\beta$	$\leqslant \beta$	$\rangle\beta$
4	82.3	17.7	12.02	9.56
5	84.3	15.7	11.76	8.15
6	85.3	14.7	11.19	7.71
7	86.3	13.7	10.94	7.29
8	86.7	13.3	10.82	7.18
9	87.1	12.9	10.77	7.09
10	87.2	12.8	10.73	7.08
11	87.5	12.5	10.67	7.05
12	87.6	12.4	10.62	7.01

According to the principle of vector synthesis, when a vector is synthesized, the longer the vector length is, the larger the inner product of the vector with the final synthesis the closer to the composite vector it is, the greater is its contribution to the composite vector. The length $\|g_i\|$ of the gradient vector g_i corresponding to the sample stands for its contribution to the total gradient; the larger the contribution $\|g_i\|$ is, the greater is its contribution to the total gradient g, that is, the greater the contribution of the sample to the change of the model parameter is. Table 1 shows the proportion of the number of samples in different sample contribution interval in the total number of samples, that is, the proportion of samples in each gradient length interval.

From Table 1, it can be seen that with the progress of the construction, the majority of the samples (about 90%) have a very small contribution to the total gradient, and only an extremely small number of samples have a relatively great influence on the total gradient. It is also found that in each construction, the intersection of sample sets with relatively small contribution to the model parameter update (for example, the sample sets with the gradient lengths from 0 to 0.1) account for more than 99%. This indicates that when the sample contributes relatively less to the total gradient in a certain construction, its contribution to the total gradient in the later construction has always been relatively small. Hence, in the whole construction calculation process, most of the samples contribute very little to the model update and convergence, and only an extremely small number of samples have a relatively great impact on the model update and convergence.

Different samples can be distinguished by the threshold value β, and when the length of the gradient vector corresponding to the sample is less than or equal to β (for $\leqslant \beta$ in Table 2), the sample is considered as having relatively little contribution to the model update; otherwise (for $> \beta$ in Table 2) it is considered that it has relatively great contribution to the model update. When $\beta = 0.08$, in each construction, the proportion of the number of samples contained in the above two types of sample sets (the sample sets that have large contribution to the model update and the sample sets that have small contribution to the model update) in the total number of samples, as well as the contribution of all samples in the sample sets to the model update (the length of gradient vector) are shown in Table 2.

From Table 2, it can be observed that when $\beta = 0.08$, more than 80% of the samples contribute very little to the model update, and the contribution of this part of the sample sets to the model update is not significantly different from the contribution of a few remaining samples to the model update.

It can be known from the above analysis that most of the samples contribute very little to the model update and the construction convergence during the construction process, and

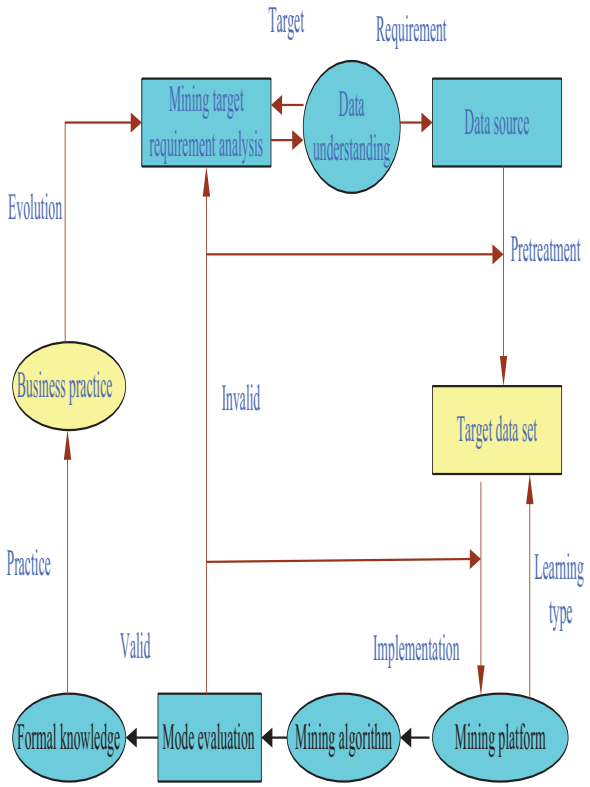

Figure 3 Mining process of the big data.

only a few samples have a relatively great impact on the model update. However, in each build, the current construction model treats each sample equally, that is, each sample is calculated once in each construction. All samples consume the same computing resources in the construction calculation process. Hence, the allocation of computing resources is not sufficiently reasonable during the whole calculation process. The samples that have relatively small impact on the results occupy a large amount of computing resources, while a small number of samples that have a relatively large impact on the results consume only a small amount of computing resources.

2.2 Java Web Big Data Mining Process

Big data refers to the set of data that cannot be captured, managed or processed by using the conventional software tools over an affordable time range. The processing of modern big data mainly includes six basic capability components: data integration, file storage, data storage, data calculation, data analysis and platform management. The big data mining process is shown in Figure 3.

In view of the unbalanced allocation of computing resources due to the current Java Web big data mining model, a construction model based on the software capability maturity model is put forward in this section, as shown in the SCMM. In this model, the computing resources are allocated according to the contribution of samples to the model update during the construction calculation process, so as to improve the utilization rate of computing resource and enhance the system performance.

SCMM Java Web big data mining process
Input: α, β, $flag \leftarrow false$
Output: Model parameter vector w.
Method: //calculate the construction calculation process of the node r in the t-th construction

(1) if $flag == false$ //if the current construction is in the rough adjustment stage;

(2) calculate the gradient of each sample based on the current model vector;

(3) else // if the current construction is in the fine-tuning stage;

(4) for each sample i;

(5) if $\left\| g_i^{(t-1)} \right\| \leqslant \beta$ // if the sample i has relatively small construction to the model update;

(6) $g_i^{(t)} \leftarrow g_i^{(t-1)}$; // use the gradient vector obtained from the last construction repeatedly;

(7) else // if the sample i has relatively large contribution to the model update in the last construction;

(8) $g_i^{(t)} = \nabla f\left(w^{(t)}, x_i, y_i\right)$; // calculate the gradient of the sample i;

(9) end if;

(10) end for;

(11) end if;

(12) calculate the total gradient g of all the samples;

(13) send the total gradient g to the server node; // the server updates the model parameter after receiving g;

(14) obtain the latest model parameter vector from the server node;

(15) if $\left\| w_r^{(t)} \leftarrow w_r^{(t-1)} \right\| \leqslant \alpha$ // the construction this time does not result in significant change to the model, and the fine-tuning stage is continued;

(16) $flag \leftarrow true$;

(17) else // if the amount of change in the model is too large, it enters the rough adjustment stage, when the gradients of all sample are calculated accurately;

(18) $flag \leftarrow false$;

(19) end if.

In the SCMM, when the sample has relatively large contribution to the model update, the exact gradient of the sample is calculated in each construction. Otherwise, in the fine-tuning stage, the gradient of the sample is not calculated while the gradient vector used in the last construction is adopted. Although the gradient vector has errors, the gradient vector has little influence on the updating and convergence of the model. In this way, the error and adverse effects caused by the reduction of calculation amount are minimized. When the error caused by the reuse of gradient vector is excessively large, the construction will enter the rough adjustment stage. At this point, it is required to calculate the accurate gradient of all the samples to ensure the correctness of the calculation results.

3. CONVERGENCE ANALYSIS

It is assumed that in the t-th construction, as the error introduced by the repeated use of the sample gradient relative to the original model is $\delta_t = O\left(\frac{1}{t}\right)$, the loss function in the optimization objective function is f, which meets the Lipschitz continuity, and the normalization function is h, then the following can be obtained from the convergence analysis of constructing the optimization model based on the software capability maturity model:

Let $\Delta_t = w_{t+1} - w_t$, then the following can be obtained at the t-th construction:

$$f(w_{t+1}) - f(w_t) \leqslant \langle \Delta_t, \nabla f(w_t) \rangle + L_{var} \|\Delta_t\|^2, \quad (1)$$

$$h(w_{t+1}) - h(w_t) \leq -\frac{1}{\gamma}\|\Delta\|^2 - \langle g_t, \Delta_t \rangle \quad (2)$$

The equation (1) and equation (2) are combined to obtain the following equation 3.

$$E\left[F(w_{t+1}) - F(w_t)\right] \leqslant \langle \Delta_t, \nabla f(w_t) - E[g_i] \rangle + \left(L_{var} - \frac{1}{\gamma}\right)\|\Delta_t\|^2, \quad (3)$$

Due to $E[g_i] = \sum_{i=1}^{m} \nabla f_i\left(w_{t_i} + \sigma_{t_i} + \delta_{t_i}\right)$, in which σ_{t_i} stands for the error in th computed node due to the delay update of the model parameter, and δ_{t_i} stands for the error caused by skipping the gradient calculation.

Let $\theta_t = O\left(\frac{1}{t}\right)$, then $\|\sigma_t\| \leqslant \theta_t, \|\delta_t\| \leqslant \theta_t$, and the following can be obtained

$$\|\nabla f(w_t) - E[g_i]\| \leqslant \sum_{k=1}^{\tau} L_{cov} \|\Delta_{t-k}\| + L_{cov}\theta_t, \quad (4)$$

Thus, the equation 5 can be obtained

$$E\left[F(w_{T+1}) - F(w_1)\right] \leqslant \sum_{t=1}^{T}\left(L_{var} + L_{cov}\tau - \frac{1}{\gamma}\right) \times \|\Delta_t\|^2 + L_{cov}\theta_t^2. \quad (5)$$

As $\theta_t = O\left(\frac{1}{t}\right)$ is established, the equation is equivalent to the convergence problem of the DBPG construction optimization model based on the software capability maturity model. Then, the software capability maturity model construction optimization strategy put forward by SCMM can ensure the convergence of the system.

4. TEST RESULTS AND ANALYSIS

In order to verify the validity of the proposed model, the logistic regression application is run on the Java Web big data mining model PS based on the RCV1 [24] and KDD04 datasets. The RCV1 dataset contains a training set of 13,000 samples and a testing set of 8,000 samples. In the test set, the KDD04 data set includes 146,000 samples. Cross-validation is adopted to select 20% of the samples as the test set, and the remaining 80% as the training set.

The test operation environment is a cluster system consisting of 4 nodes, each computer has the Intel Xeon4 core CPU with a frequency of 2.4 GHz. The cluster nodes are connected by 10 Gbps Ethernet. In the four nodes, one is used as the scheduler node, one as the server node and two as the computing nodes.

The logistic regression problem is taken as an example to study the above model on the RCV1 and KDD04 datasets. The difference in the total calculation amount is constructed and the influence of the above model on the accuracy of the training model is evaluated, which is then compared with the latest big data mining model DBPG at present.

Figure 4 shows the comparison of the total computation amount between the SCMM and DBPG models on the RCV1 and KDD04 datasets. On the RCV1 dataset, the SCMM model requires construction for 18 times before convergence. However, the gradients of all samples need to be calculated in only 4 times. In the rest of the constructions, it is only required to calculate the gradient of a few samples. Compared with the DBPG model, the total computational amount of the model is reduced by 35%. On the KDD04 dataset, the SCMM model requires construction for 15 times before convergence. However, the gradients of all samples need to be calculated in only 4 times. In the rest of the 11 constructions, it is only required to calculate the gradient of a few samples. Compared

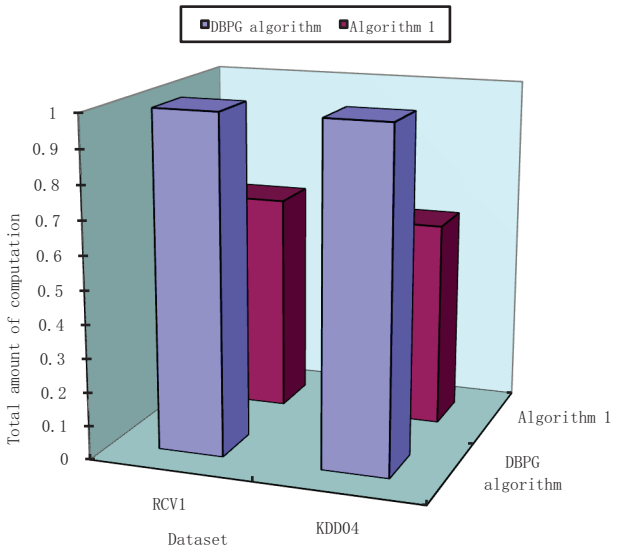

Figure 4 Comparison of the total amount of computation between the SCMM model and the DBPG model.

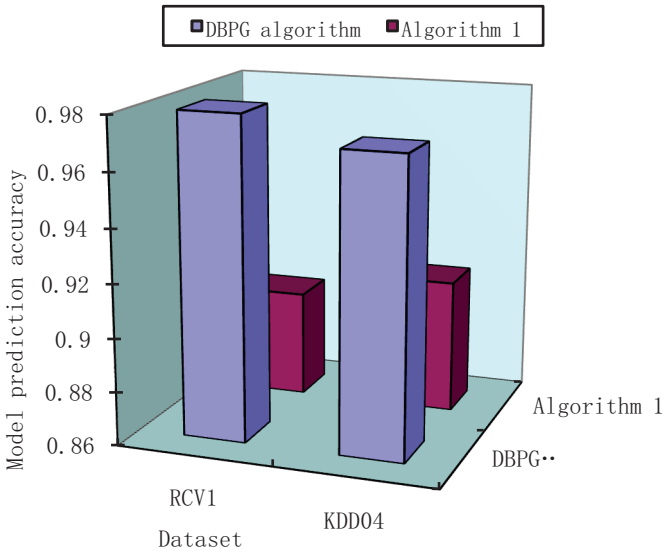

Figure 5 Comparison of the model prediction accuracy between the SCMM model and the DBPG model.

with the DBPG model, the total computational amount of the model is reduced by 38%.

Figure 5 shows the comparison of the SCMM and DBPG model training accuracy on the RCV1 and KDD04 datasets. On the RCV1 dataset, the SCMM training model has a prediction accuracy of approximately 90.5% for the test set, while the DBPG training model has a prediction accuracy of about 97.3%. On the KDD04 dataset, the SCMM training model has a prediction accuracy of about 92.4% for the test set, while the DBPG training model has a prediction accuracy of about 98.2%.

From Figure 4 and Figure 5, it can be known that the SCMM model can greatly reduce the amount of computation in the execution process of machine learning optimization model construction, and the effect on model accuracy can be controlled within a certain range. Given that the application of Java Web big data mining model has much higher requirements for the real-time and rapidity performance of computation than those for the model accuracy, it is demonstrated that the SCMM model is more suitable for big data mining than the mainstream big data mining DBGP model at present.

In order to increase the data level of the data set and achieve the desired experimental effect, the repetition multiplication principle is applied to expand the volume of the original data set and maximize the data volume. This experiment is carried out on the Java Web big data mining model experimental platform based on the software capability maturity model. The experimental process is as follows: the volume of the target data is fixed, and different minimum support degrees are set to study the time efficiency issue of the two algorithms in obtaining the frequent item sets. The experimental results are obtained by recording the experimental data as shown in Figure 6.

From Figure 6, it can be known that the efficiency of the SCMM algorithm is superior to that of the DBPG algorithm, especially when the minimum support threshold value is smaller, the execution efficiency of the improved SCMM

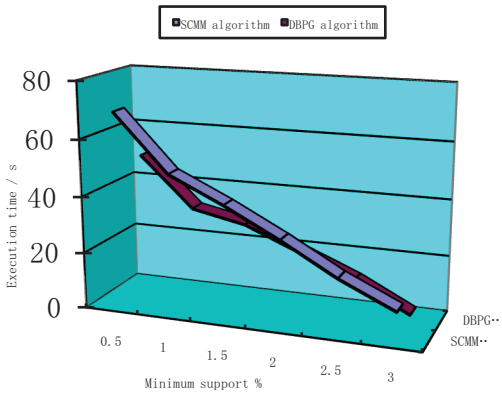

Figure 6 Execution time of the model.

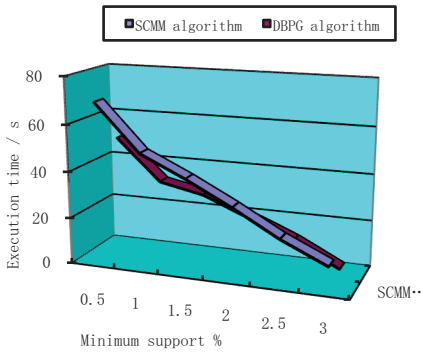

Figure 7 Scalability of the model.

algorithm is significantly increased. Subsequently, in order to study the efficiency of the algorithm under different data volumes, further research experiments are carried out. The minimum support is fixed to 1% (the minimum support is reduced, and the calculation time of the algorithm is increased as much as possible), and different target data volumes are set to study the time efficiency issue of the two algorithms in obtaining the frequent item sets. The experimental results are obtained by recording the experimental data as shown in Figure 7.

From Figure 7, it can be known that the SCMM algorithm has higher efficiency with the increase of the data volume within a certain range. As the data volume is increased within a certain range, the execution time of the DBPG algorithm is increased and the efficiency is decreased. The above experimental results have shown that SCMM parallelization is effective and feasible, which is superior to the non-parallelized SCMM. At the same time, it has indirectly verified the feasibility of the Java Web big data mining model.

5. CONCLUSIONS

This study proposes to apply the application characteristics of Java Web big data mining model to the construction of the optimization model for the distributed machine learning. Compared with the traditional big data mining models, this model can achieve a significant increase in the system execution efficiency at the cost of a small amount of model accuracy. The proposed model mainly makes use of the feature that most of the construction leads to relatively small change in the model during the construction execution process of the current optimization model. In addition, most of the samples in the calculation process contribute very little to the model update. Therefore, the sample gradient with relatively small contribution is used repeatedly in the later stage of model update to reduce the amount of computation consumed by this part of samples, thereby improving the system execution efficiency.

REFERENCES

. Zhang Y., Guo S. L., Han L. N., Li, T. L. (2016): Application and exploration of big data mining in clinical medicine. Chinese Medical Journal. vol. 129, no. 6, pp. 731–738.

1. Weichselbraun, A., Gindl, S., & Scharl, A. (2014). Enriching semantic knowledge bases for opinion mining in big data applications. Knowledge-Based Systems, 69, 78–85.
2. Yue, Z., Shuli, G., Lina, H., & Tieling, L. . (2016). Application and exploration of big data mining in clinical medicine. Chinese Medical Journal, 129(6), 731–738.
3. Xu, L., Jiang, C., Chen, Y., Wang, J., & Ren, Y. . (2016). A framework for categorizing and applying privacy-preservation techniques in big data mining. Computer, 49(2), 54–62.
4. Fong, S., Liu, K., Cho, K., Wong, R., Mohammed, S., & Fiaidhi, J. (2016). Improvised methods for tackling big data stream mining challenges: case study of human activity recognition. Journal of Supercomputing, 72(10), 3927–3959.
5. (2016). Recent advances in social multimedia big data mining and applications. Multimedia Systems, 22(1), 1–3.

6. Liu, Z. . (2015). Fast kernel feature ranking using class separability for big data mining. The Journal of Supercomputing, 72(8), 3057–3072.
7. Zhu, L., Li, M., Zhang, Z., Du, X., & Guizani, M. (2018). Big data mining of users' energy consumption patterns in the wireless smart grid. IEEE Wireless Communications, 25(1), 84–89.
8. Grammer, A. C., Ryals, M. M., Heuer, S. E., Robl, R. D., Madamanchi, S., & Davis, L. S. (2016). Drug repositioning in sle: crowd-sourcing, literature-mining and big data analysis. Lupus, 25(10), 1150–1170.
9. Wang, & Zhe. (2017). Big data mining powers fungal research: recent advances in fission yeast systems biology approaches. Current Genetics, 63(3), 427–433.
10. Macfadden, B. J., & Guralnick, R. P. (2017). Horses in the cloud: big data exploration and mining of fossil and extant equus (mammalia: equidae). Paleobiology, 43(01), 1–14.
11. Luna, J. M., Padillo, F., Pechenizkiy, M., & Ventura, S. (2017). Apriori versions based on mapreduce for mining frequent patterns on big data. IEEE Transactions on Cybernetics, 1–15.
12. Glatz, E., Mavromatidis, S., Ager, B., & Dimitropoulos, X. (2014). Visualizing big network traffic data using frequent pattern mining and hypergraphs. Computing, 96(1), 27–38.
13. Schl?Tterer, C., Tobler, R., Kofler, R., & Nolte, V. (2014). Sequencing pools of individuals — mining genome-wide polymorphism data without big funding. Nature Reviews Genetics, 15(11), 749–763.
14. Ma, S., Li, J., Liu, L., & Le, T. D. (2016). Mining combined causes in large data sets. Knowledge-Based Systems, 92, 104–111.
15. Lin, C. W., & Hong, T. P. (2014). Maintenance of prelarge trees for data mining with modified records. Information Sciences, 278, 88–103.
16. Eldawlatly, S., Jin, R., & Oweiss, K. G. (2009). Identifying functional connectivity in large-scale neural ensemble recordings: a multiscale data mining approach. Neural Computation, 21(2), 450–477.
17. Robertson, S. P., Quon, H., Kiess, A. P., Moore, J. A., Yang, W., & Cheng, Z. (2015). A data-mining framework for large scale analysis of dose-outcome relationships in a database of irradiated head and neck cancer patients. Medical Physics, 42(7), 4329–4337.
18. Lin, C. W., & Hong, T. P. . (2014). Maintenance of prelarge trees for data mining with modified records. Information Sciences, 278, 88–103.
19. Caldwell, P. M., Bretherton, C. S., Zelinka, M. D., Klein, S. A. , Santer, B. D., & Sanderson, B. M. (2014). Statistical significance of climate sensitivity predictors obtained by data mining. Geophysical Research Letters, 41(5), 1803–1808.
20. Kim, K. H., Lee, S., Shim, J. B., Chang, K. H., Yang, D. S., & Yoon, W. S. (2017). A text-based data mining and toxicity prediction modeling system for a clinical decision support in radiation oncology: a preliminary study. Journal of the Korean Physical Society, 71(4), 231–237.