A Dynamic Convergence Criterion for Fast K-means Computations

Hui Yu[1], Yujie Du[2], Xiaoqi Zhang[1], Zhigang Wang[1(✉)], Ning Wang[1], Juncheng Yi[3], Xiaodong Wang[1], Jie Nie[1], and Zhiqiang Wei[1]

[1] Ocean University of China, Qingdao, China
yuhui1472@stu.ouc.edu.cn,
{wangzhigang,wangning8687,wangxiaodong}@ouc.edu.cn
[2] Yantai Engineering and Technology College, Yantai, China
[3] Big Data Center, Qingdao, China
yijuncheng@qd.shandong.cn

Abstract. The K-Means algorithm has effectively promoted the development of intelligent systems and data-driven decision-making through data clustering and analysis. A reasonable convergence judgment directly determines when the model training can be terminated, which heavily affects the model quality. There are many researches for training acceleration and quality improvement, but few focus on the judgment. Currently, the convergence criteria still adopt a centralized judgment strategy for a single loss value. The criterion is simply copied between different optimized K-Means variants, typically like the fast Mini-Batch version and the traditional Full-Batch version. Our analysis reveals that such a design cannot guarantee that different variants converge to the same point, that is, it can result in abnormal situations such as false-positive and over-training. To perform a fair comparison and guarantee the model accuracy, we proposed a new dynamic convergence criterion VF (Vote for Freezing) and optimized version VF+. VF adopts a distributed judgment strategy where each sample can decide whether to participate in training based on the criterion (i.e., freezing itself) or not. Meanwhile, combined with the priority of samples, VF adaptively adjusts the sample freezing threshold which achieves asymptotic withdrawal of samples and accelerates model convergence. VF+ further introduced parameter freezing thresholds and freezing periods to eliminate redundant distance calculations, hence it improves the training efficiency. Experiments on multiple datasets validate the effectiveness of our convergence criterion in terms of training quality and efficiency.

Keywords: K-Means · Convergence Criterion · Priority

1 Introduction

As the most widely used clustering algorithm [1,4], the K-Means algorithm has been extensively explored. Several methods have been proposed to enhance

C. Jin et al. (Eds.): WISA 2024, LNCS 14883, pp. 191–202, 2024.
https://doi.org/10.1007/978-981-97-7707-5_17

clustering quality [9]. Besides, given the K-Means algorithm's high dependency on centroid initialization, optimizing the initial centroid is a good strategy to improve the model convergence speed [18]. The Canopy algorithm proposed by McCallum et al. [8] improved distance calculation and Pérez et al. [10] enhanced classification. Parallel computing and distributed algorithms [7,14] improved the efficiency of processing large-scale data.

The above optimization methods are all applied to two basic training policies: Full-Batch and Mini-Batch. Compared with Full-Batch, where centroids are updated after all samples are scanned, Mini-batch can perform frequent updates when partial samples are processed. Mini-Batch speeds up convergence because fresh information learned from previously processed samples can be quickly used to process subsequent samples. Nowadays, Mini-Batch has become a popular choice for model training.

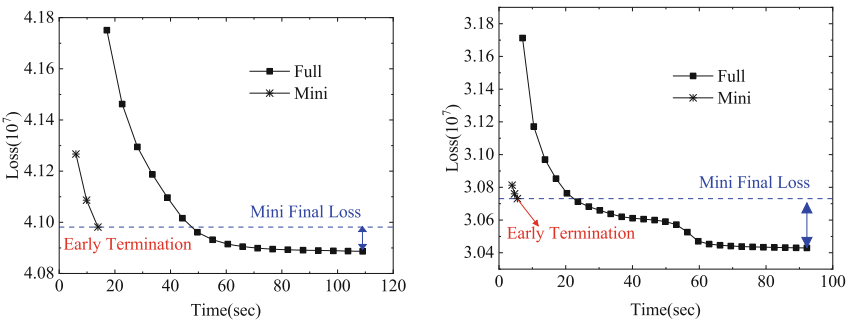
The training process iteratively adjusts parameters until convergence. There are multiple convergence criteria including reaching a pre-set number of iterations, minimizing loss below a threshold, or finding the loss difference between two consecutive iterations below a given threshold. The thresholds of the first two criteria are difficult to set, and it is difficult to ensure model quality. The most commonly used convergence criterion is the last one.

Currently, when running the Mini-Batch training, researchers usually directly use the loss difference threshold in Full-Batch. However, such a design can lead to unfair comparisons. Under Mini-Batch, the accumulated change of the loss function per iteration is smaller than that in Full-Batch because the former only processes partial samples instead of all. Then, the difference threshold used in Full-Batch can be easily satisfied under Mini-Batch. This false-positive judgment will terminate training abnormally. A possible solution is normally updating the centroid after each Batch but only judging convergence after completing an Epoch (resembling Full-Batch). However, such a delayed judgment detection may result in redundant training, because the frequent update will largely change the centroids during an epoch. The loss difference between two epochs can be overly large, making it difficult to satisfy the threshold (but actually the absolute loss value has been significantly smaller than that achieved in Full-Batch).

We use two real-world datasets HIGGS and HepMASS to validate the false-positive and over-training phenomena on K-Means. Figure 1 shows the phenomenon of false-positive convergence. Table 1 shows the phenomenon where over-training generates 13.2% and 17.9% runtime degradation.

In order to solve the above problems, we introduce a new threshold-setting policy based on the "Vote-to-Halt" [6] mechanism and study a new convergence algorithm VF. The reason for the two phenomena is that the convergence judgment always resorts to the accumulated loss value. Such a centralized policy is very sensitive to the number of samples involved in the computation. Our solution is designing a decentralized policy where each sample can individually decide whether to participate in training (i.e., it can vote to halt if it's unnecessary to be continuously processed) or not.

Recently, researchers also noted that some non-critical samples do not need to participate in training models. The SlimML [5] framework eliminates non-critical data during iterations. It selects a small number of representative aggregated data samples based on a priority threshold for training. However, its prioritization method can't be directly applied to K-Means due to the absence of gradient computation. Besides, The threshold for excluding non-critical data is set manually, and each sample shares the same constant threshold. We change the prioritization method for K-Means and give a dynamic threshold.



(a) HIGGS (b) HepMASS

Fig. 1. Evaluation of false-positive. The blue dashed line represents the final loss value of the Mini-Batch when the Full-Batch threshold is directly used. (Color figure online)

Table 1. Evaluation of over-training. Truncated Time is the time when the Mini-Batch loss descends to match the Full-Batch for the first time. Complete Time is the final training time of the Mini-Batch.

Datasets	Full-Batch Loss(10^7)	Mini-Batch Loss(10^7)	Truncated Time(sec)	Complete Time(sec)
HepMASS	3.0429643	3.0426370	39.272	45.236
HIGGS	4.0886614	4.0883565	43	52.364

In addition to sample-level freezing, we are aware that some work also focuses on parameters, i.e., Adaptive Parameter Freezing (APF) [3], which avoids high communication costs in distributed environments by freezing stable parameters. The freezing period is adjusted using the TCP approach based on the stability of unfrozen parameters. We also borrow its idea to accelerate K-Means but the frozen threshold and loss effective variation requires careful design.

Inspired by "Vote-to-Halt" and APF, we propose new convergence criteria VF and VF+. Our main contributions can be summarized as follows:

- We propose a novel convergence criterion (VF) based on a distributed loss metric. This criterion replaces the commonly used centralized loss metric to eliminate avoid the unfairness caused by directly transferring loss thresholds from Full-batch to Mini-Batch.
- We propose a priority-based adaptive sample freezing strategy. We develop a simple but efficient prioritization strategy based on the contribution of each data sample to convergence. To boost model convergence efficiency, we gradually withdraw samples by dynamically adjusting the freezing threshold.
- An adaptive parameter freezing strategy for K-Means (VF+) is proposed. This strategy dynamically adjusts parameter freezing thresholds and freezing periods based on the characteristics of each parameter. By minimizing redundant sample-centroid distance calculations, we reduce model training time.

2 Method

In this section, we will provide a detailed explanation of our proposed new convergence criteria, VF and VF+.

A detailed introduction to VF will be given in Sect. 2.1. VF is a decentralized convergence criterion which samples with low priority are withdrawn from training by voting. Section 2.2 will provide a detailed introduction to VF+. VF+ builds upon VF, adding a parameter freezing module.

2.1 VF: "Vote-to-Halt" for Samples

VF is a sample voting freeze algorithm that combines the "Vote-to-Halt" with the freeze threshold for samples.

A. Vote-to-Halt. There are several graph processing systems [11–13,15] and Vote-to-Halt is a part of the Pregel. In Pregel, each vertex has a binary identifier *vote* which identifies its two states: active and inactive (i.e., participating in computation or not). After all vertices vote their status as inactive, the graph algorithm terminates (converges). We treat samples in K-Means as vertices in the graph. In each iteration, these samples independently decide whether to join the current training or not. This approach differs from the traditional method where samples either fully participate or abstain from the computation.

VF sets a state, $vote_n$, to sample X_n. X_n also has two states: active and inactive, changing dynamically during model training. Initially, all of the samples are active and participate in iteration calculations. Every sample is assigned a freezing threshold based on the Full-Batch threshold. If the point's loss-difference between two consecutive iterations falls below this threshold, the point will make little contribution to the model and the state of the sample will switch to inactive.

VF localizes the "Vote-to-Halt" technique specifically for K-Means. On the one hand, VF does not adopt the mechanism of active messages. These messages

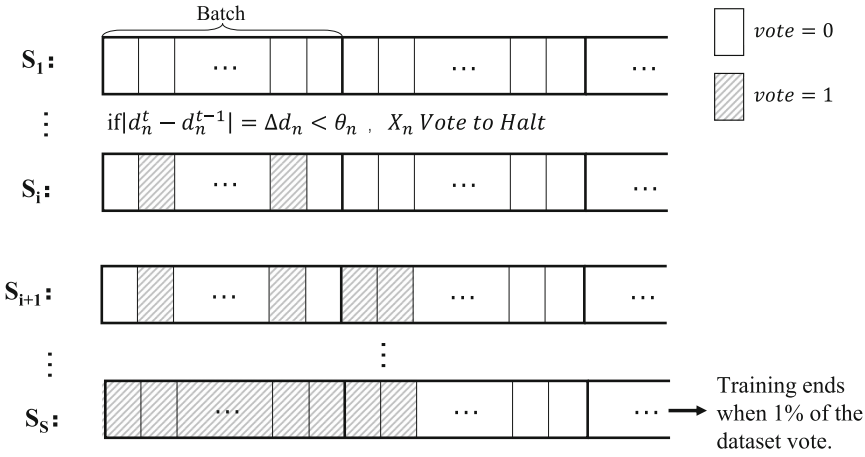


Fig. 2. Overall framework of "Vote-to-Halt" applied to K-Means

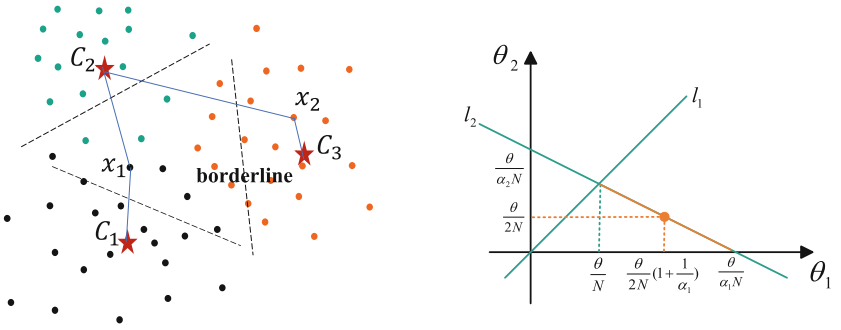
are only sent when the distance of the sample to the centroid changes significantly. Monitoring this requires checking the distance of each sample to the centroid at each iteration, regardless of convergence. Even though this approach may enhance convergence quality, it brings substantial computational costs. Experimental evaluation reveals that using messages only results in a quality improvement of 0.04% to 0.09%, with a time loss ranging from 11% to 64%.

On the other hand, we think the model converged when 1% of the samples have voted rather than all. Initially, with significant centroid variations, only a few samples vote, but the loss function fluctuates greatly. As the training progresses, centroids stabilize, leading to decreased distances between samples and centroids, thereby lowering the likelihood of cluster changes. After the loss function is stable, continuing to require all samples to converge will not greatly improve the accuracy of the model, so redundant training should be stopped in time. The voting framework of VF is illustrated in Fig. 2.

B. Freezing Threshold of Sample. The freezing threshold of sample is a crucial parameter. Next, we will provide a detailed explanation of how to determine the threshold. Simply, we can evenly distribute the Full-Batch threshold θ to all samples. However, using this threshold might not fully leverage the individual characteristics of each sample. Some samples might not exit at the right time, leading to inefficient resource usage and low training accuracy. Therefore, we introduce sample priorities to adjust the thresholds.

The distance from the sample to its centroid C_1 is d_1. The next closest centroid is C_2, and the distance is d_2. In most of time, sample change its centroid from C_1 to C_2. If the difference between d_1 and d_2 is not much large, it means that the sample is very easy to change cluster. These samples are labeled as high-priority due to their potential to impact model parameters. Others are labeled

as low-priority. As shown in Fig. 3(a), x_1 is a high-priority sample while x_2 is a low-priority sample.



(a) Illustration of sample priority

(b) Illustration of threshold for different priority samples

Fig. 3. Illustration of priority.

To classify the samples, the key is to observe R, the ratio of d_2 to d_1. If R is less than the threshold α, it means that the gap between d_1 and d_2 is large and it is stable. Whether the α is reasonable or not directly determines the effectiveness of the method. The changes in the centroid gradually become stable, so α also should be reduced. The threshold of the j^{th} iteration we designed is the mean of all sample ratios R in the previous epoch. N is the number of samples in the dataset:

$$\alpha_j = \frac{1}{N}\sum_{n=1}^{N} R_n^{j-1} \tag{1}$$

The freezing threshold assignment strategy should satisfy two conditions. First, samples with higher priority have lower freezing thresholds, which makes high-priority samples participate in more training. Second, the sum of sample thresholds must equal the Full-Batch threshold. The freezing thresholds should not be increased or decreased blindly. Otherwise, it will lead to underfitting and overfitting. We define the freezing threshold for low-priority samples as θ_1 and for high-priority samples as θ_2. To satisfy the above conditions, the constraints given by the thresholds are as follows:

$$s.t. \begin{cases} \theta_1\varphi_1 N + \theta_2\varphi_2 N = \theta \\ \theta_1 > \theta_2 \\ \theta_1,\ \theta_2 > 0 \\ \varphi_1 + \varphi_2 = 1 \end{cases} \tag{2}$$

Here, φ_1 represents the proportion of low-priority samples, and φ_2 represents the high-priority. θ is the Full-Batch threshold. According to Fig. 3(b), l_1 is the

second expression in Eq. 2. l_2 is the first expression in Eq. 2. It is easy to see that the orange line satisfies the condition, thus we have:

$$\begin{cases} \theta_1 = \dfrac{\theta}{N}\ (1 - k + \dfrac{k}{\alpha_1}) \\ \theta_2 = (1 - k)\dfrac{\theta}{N} \end{cases} \tag{3}$$

k is used to control the difference between θ_1 and θ_2. Generally, we set k=0.5.

2.2 VF+: Freezing for Parameters

Inspired by APF, a parameter freezing module is added to VF to further improve the training efficiency, forming the VF+ algorithm which is shown in Fig. 4.

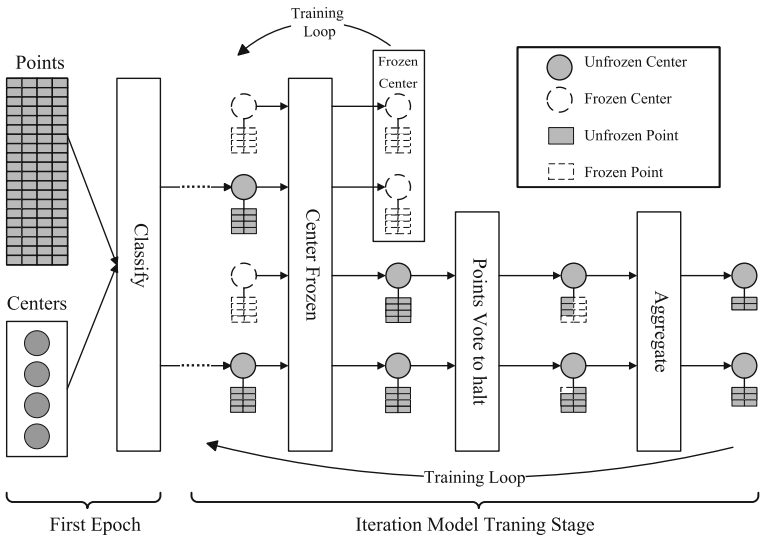


Fig. 4. Overview of VF+. Using four centroids as an example: After centroids are frozen, they can unfreeze during training, whereas samples, once frozen, will no longer be involved in calculations

In K-Means, the parameters of the model are the centroids. When the centroid undergoes slight changes, distances between samples and their centroids will not change greatly. We greedily think that few other centroids will undergo significant changes, which means the probability of centroid change is extremely low. Hence, it is of little significance to calculate sample points in clusters with smaller centroid changes. Especially, when large-scale datasets are applied to K-Means, there could be tens or even millions of samples in a cluster, and each iteration needs to calculate the distance of each sample point to all centroids. Therefore, freezing certain centroids and their samples at appropriate times can

significantly reduce redundant distance calculations. However, frozen parameters must be unfrozen at the right time to maintain quality clustering. Although this may cause a loss in convergence accuracy, experiments show that the time saved outweighs this loss.

A. Frozen Threshold & Period. This section introduces how we apply some methods in APF to K-Means. Similar to samples, centroid C_i gradually stabilizes with iterations, requiring the freezing threshold ε_i^r to be dynamically adjusted. Freezing all parameters makes the mechanism useless, yet leaving all unfrozen would lack centroids for training. At first, we give an initial parameter freezing threshold. If most parameters are frozen, we halve the threshold to reduce the likelihood of all parameters being frozen. The freezing period also needs careful consideration. Too short, it will risk redundant detection; too long, it will slow convergence. Hence, we use TCP mode to adaptively adjust the freezing period. The parameters are tracked after they are unfrozen. If a parameter is frozen again, its threshold will be increased by 1. If they remain unfrozen, we greedily halve the freezing period.

B. Loss Function's Effective Variation. In APF, a parameter is a single value, making it easier to calculate positive and negative directions for effective perturbation. However, in K-Means, a parameter is a multi-dimensional vector, making it challenging to compute positive and negative directions. If we are to calculate the positive and negative directions of the centroids for each dimension independently, some centroid dimensions will be frozen, leading to biased clustering. Hence, in K-Means, we focus on the Euclidean distance between centroid positions to determine parameter variation. If the distance P_n^r is less than a threshold ε_i^r, the centroid is frozen.

$$P_n^r = \|C_n^r - C_n^{r-1}\|_2, (X_n \in C_i)$$

3 Experiments

In this section, we will present the experimental results of VF and VF+ on three datasets. In Sect. 3.1, we will introduce the datasets and our baseline. In Sect. 3.2, we will present a comparison between our methods and baselines to demonstrate the effectiveness of our approaches. In Sect. 3.3, we will conduct experiments to analyze the necessity of certain details in our method.

3.1 Experimental Setup

A. Datasets. The details of the three datasets we used are shown in Table 2.

Table 2. Dataset Summary

Datasets	Samples Size	Dimensions
Covtype [2]	581012	54
HIGGS [16]	11000000	28
HepMASS [17]	7000000	27

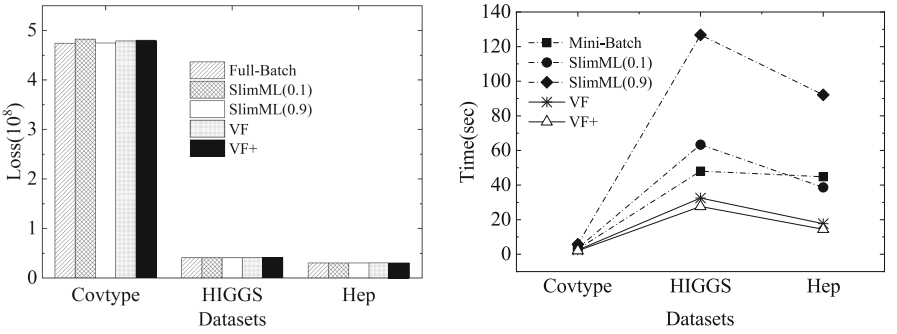
B. Baseline. We compare the models in terms of both accuracy and training time. In terms of model accuracy, our intention is to use SlimML as the baseline. However, the freezing threshold for samples in SlimML is selected through many experiments, and exact values for K-Means are not given. Therefore, we use two extreme values $\theta/(k * N)$, with $k = 0.1$ and $k = 0.9$ as the baseline. But users typically need the Full-Batch training loss, so we add a Full-Batch baseline, just in case. In terms of training time, we also use the two extreme values of SlimML and an adding one as the baseline. However, directly applying the Full-Batch threshold to Mini-Batch training will lead to unfairness. Therefore, it is not wise to directly use the training time of these two ways as baselines. In Mini-Batch training mode, we use the truncated time as the time baseline.

3.2 Overall Performance

Figure 5(a) shows the results of comparing the loss and runtime among VF, VF+, and baselines. For these three datasets, the maximum difference between VF and the three loss baseline is 0.97%, 0.05%, and 0.59%. The maximum difference between VF+ and the three loss functions baseline is 1.26%, 0.06%, and 0.73%. It can be seen that the loss value of our method and the three baselines can be bounded. Figure 5(b) shows that our method achieves a significant time gain compared to all three baseline methods on the HIGGS and HepMASS datasets. Due to the different orders of magnitude, the difference in Covtype is not fully reflected in Fig. 5(b). For VF, Covtype reduces the time by 18.61% over Full-Batch and by 31.65% and 58.26% over SlimML. For VF+, the time of Covtype is 30.55% shorter than Full-Batch, 41.68% and 64.38% shorter than SlimML.

3.3 Analysis of VF & VF+

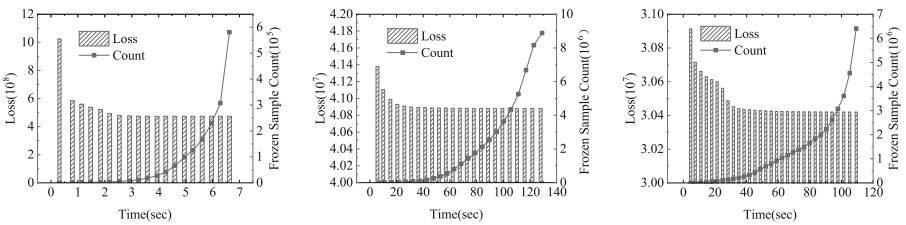
A. Dynamic Sample Freezing Threshold. As iterations progress, centroids stabilize, leading to stabilized distances from samples. Therefore, it is evidently unreasonable to use a fixed sample freezing threshold and we have confirmed this suspect through experiments. We set the fixed sample freezing threshold as $\theta/(k * N)$, where k is given as 0.1, 0.3, 0.5, 0.7, and 0.9. Model accuracy and training time trend in opposite directions as k grows, and there is no satisfactory threshold for model accuracy and training time.



(a) Comparison of loss in different datasets (b) Comparison of time in different datasets

Fig. 5. Overall performance.

B. Partial Samples Converge. In Sect. 2.1, we have given a detailed explanation for adopting partial samples convergence. Here we give specific experimental validation. Figure 6 illustrates the running results on VF but lets all samples converge across three datasets. The bar is the loss and the line is the frozen sample count. We can see that in the early stage of training, only a few samples are frozen, while the loss makes a huge change. In the later stages, although the number of frozen samples increases, the change of loss is almost negligible. so we truncate at 1% on the three datasets, and we get a loss function that differs from the fully converged loss function by only 1.06%, 0.03%, and 0.60%. But we get 60.72%, 71.94%, and 81.42% gains in time.

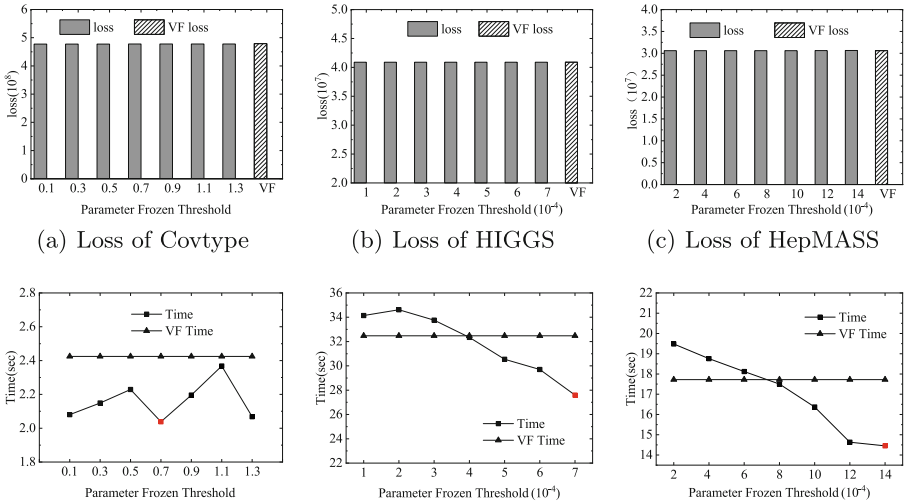


(a) Covtype (b) HIGGS (c) HepMASS

Fig. 6. Changing process of loss and frozen sample count with the entire dataset converged

C. Parameter Frozen Threshold. Since K-Means has fewer parameters, we've found through experiments that if the threshold is too large, all parameters may be frozen and our halving strategy won't have time to take effect. Consequently, samples will idle until a parameter is unfrozen. This will significantly weak the training efficiency. To find a better threshold, we gradually reduce the freezing threshold of the parameters while ensuring that no idling occurs.

We conduct experiments with 7 thresholds for each dataset, and the results are shown in Fig. 7. The loss value of model training changes relatively little, so we select the freezing thresholds with the shortest running time, which are 0.7, 0.0007, and 0.0014.



(a) Loss of Covtype (b) Loss of HIGGS (c) Loss of HepMASS

(d) Time of Covtype (e) Time of HIGGS (f) Time of HepMASS

Fig. 7. Loss and training time with different freezing thresholds for parameters.

4 Conclusion

VF uses decentralized convergence and sample priority based on distance ratios to address unfairness when applying the loss threshold of Full-Batch to Mini-Batch. VF+ enhances efficiency with dynamically freezing parameter freezing. Experiments show that VF and VF+ improved efficiency and ensured quality.

Acknowledgement. This work was supported by the Key R&D Program of Shandong Province, China (No. 2023CXPT020), and the National Natural Science Foundation of China (No. U23A20320 and No. U22A2068).

References

1. Ahmed, M., Seraj, R., Islam, S.M.S.: The k-means algorithm: a comprehensive survey and performance evaluation. Electronics **9**(8), 1295 (2020)
2. Blackard, J.: Covertype. UCI Machine Learning Repository (1998). https://doi.org/10.24432/C50K5N
3. Chen, C., et al.: Communication-efficient federated learning with adaptive parameter freezing. In: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), pp. 1–11. IEEE (2021)

4. Ghazal, T.M.: Performances of k-means clustering algorithm with different distance metrics. Intell. Autom. Soft Comput. **30**(2), 735–742 (2021)
5. Han, R., et al.: SlimML: removing non-critical input data in large-scale iterative machine learning. IEEE Trans. Knowl. Data Eng. **33**(5), 2223–2236 (2019)
6. Malewicz, G., et al.: Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 135–146 (2010)
7. Mao, Y., Gan, D., Mwakapesa, D.S., Nanehkaran, Y.A., Tao, T., Huang, X.: A mapreduce-based k-means clustering algorithm. J. Supercomput. 1–22 (2022)
8. McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 169–178 (2000)
9. Olukanmi, P., Nelwamondo, F., Marwala, T., Twala, B.: Automatic detection of outliers and the number of clusters in k-means clustering via Chebyshev-type inequalities. Neural Comput. Appl. **34**(8), 5939–5958 (2022)
10. Pérez, J., Martínez, A., Almanza, N., Mexicano, A., Pazos, R.: Improvement to the k-means algorithm by using its geometric and cluster neighborhood properties. In: Proceedings of ICITSEM, pp. 21–26 (2014)
11. Song, Z., et al.: ADGNN: towards scalable GNN training with aggregation-difference aware sampling. Proc. ACM Manag. Data **1**(4), 1–26 (2023)
12. Song, Z., Gu, Y., Qi, J., Wang, Z., Yu, G.: EC-graph: a distributed graph neural network system with error-compensated compression. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp. 648–660. IEEE (2022)
13. Wang, Z., Gu, Y., Bao, Y., Yu, G., Yu, J.X., Wei, Z.: HGraph: I/O-efficient distributed and iterative graph computing by hybrid pushing/pulling. IEEE Trans. Knowl. Data Eng. **33**(5), 1973–1987 (2019)
14. Wang, Z., et al.: FSP: towards flexible synchronous parallel frameworks for distributed machine learning. IEEE Trans. Parallel Distrib. Syst. **34**(2), 687–703 (2022)
15. Wang, Z., et al.: Lightweight streaming graph partitioning by fully utilizing knowledge from local view. In: 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), pp. 614–625. IEEE (2023)
16. Whiteson, D.: HIGGS. UCI Machine Learning Repository (2014). https://doi.org/10.24432/C5V312
17. Whiteson, D.: HEPMASS. UCI Machine Learning Repository (2016). https://doi.org/10.24432/C5PP5W
18. Yu, C., Fei, L., Chen, F., Chen, L., Wang, J.: Heterogeneous graphs embedding learning with metapath instance contexts. In: Yuan, L., Yang, S., Li, R., Kanoulas, E., Zhao, X. (eds.) WISA 2023. LNCS, vol. 14094, pp. 149–161. Springer, Singapore (2023). https://doi.org/10.1007/978-981-99-6222-8_13