Reading Report for FastSGD Paper

Chenxi Huang

I. Introduction

FastSGD is a fast compressed SGD framework for distributed machine learning[1]. In the data parallelism of distributed machine learning, stochastic gradient descent runs in a way where data are distributed to every nodes and gradients are calculated independently, which will be aggregated on prior model parameters to produce next iteration of the model. While parallel computing for the model is achieved, the problem that the communication cost emerges when it comes to the convergence of models with more parameters. Researchers proposed FastSGD to compress the gradient information to implement data parallelism more efficiently. In this reading report, I summarize the major improvement made by FastSGD compared to prior work SketchML, and do calculation to present the performance of FastSGD in certain scenarios, and finally, I propose my ideas about further development on FastSGD.

II. Major Improvement: Compared with SketchML

Similar to SketchML, the high performance of FastSGD is based on the hybrid method of quantization methods and sparsification methods to compress gradient key/value pairs. On the one hand, the quantization methods provide a lossy approach to compress the gradient value. On the other hand, sparsification methods reduce the number of parameters based on the assumption that a large number of gradient values are contributing less to the convergence of the model. With these two types of methods, FastSGD and SketchML are able to compress gradient data dramatically.

Compared to SketchML, FastSGD[1] proposes several improvement to outperform contemporary algorithms including SketchML.

A. Improvement on Compression of Gradient Values

Like SketchML, FastSGD performs compression on gradient keys and values. For the compression of gradient values, SketchML performs data sketch and other techniques to perform lossy compression, while FastSGD maps gradient values through reciprocal mapper, does logarithm quantization on it, and filter parameters within the threshold. Operations in transformation of gradient values mentioned above are mathematically fast. The time complexity of the FastSGD encoding is linear, faster than SketchML, achieving high performance gradient value compression.

B. Improvement on Compression of Gradient Keys

FastSGD performs lossless compression of gradient keys as same as SketchML, while FastSGD implements this algorithm by processing gradient keys utilizing finer grain(per bit instead of per byte in SketchML) and adaptive length flags. The experiment[1] shows the fact that reduced redundant bits contribute to higher performance in lossless gradient key compression.

III. FastSGD Performance in Certain Scenarios

A. Given the case that the maximal delta key is 255, length flag is 2 bits long, delta keys are evenly distributed, what is the number of bits for 400 compressed gradient key?

The expected space for a compressed gradient key is:

$$\frac{1}{256}(3+\sum_{l=1}^{8} 2+l\times 2^{l-1}) = \frac{1812}{256} = \frac{453}{64} \approx 7.078125$$

Since the compression of the gradient key is independent[1], the overall expected space is 1812-bits long.

B. Theoretic Compression Performance of FastSGD

In FastSGD, a gradient value is reciprocal mapped, logarithm operated and threshold filtered. The size of the final outcome depends on the hyperparameter τ, which restricts the logarithm operated gradient value to be below τ[1]. Therefore, the size is reduced to $\lceil 1+\log_2 \tau \rceil$ (the extra 1 is for the sign bit).

For a gradient value of length l, the compression ratio is:

$$\frac{l}{\lceil 1+\log_2 \tau \rceil}$$

Theoretic compression ratio for usual configurations are calculated and presented as TABLE I.

TABLE I
Theoretic Compression Ratio

Ratio	$\tau = 32$	$\tau = 64$	$\tau = 128$	$\tau = 256$
32-bit	5.33	4.57	4.00	3.55
64-bit	10.66	9.14	8.00	7.11
128-bit	21.33	18.29	16.00	14.22

IV. Ideas about Further Development on FastSGD

FastSGD has achieved high performance with reduced time cost and promoted compression ratio. However, I have some ideas about how to improve this work.

A. Experiment on Popular Deep Learning Models

In the original paper[1], only generalized linear models, SVM and basic neural network architectures(multi-layer perceptron) are tested. I propose the experiment on convolutional neural network(CNN), Transformer and other popular deep learning models is significant to distributed machine learning.

B. Optimization of Gradient Key Compression

In FastSGD, the gradient key is compressed individually by the delta key and the length-flag adaptation mechanism. I suspect that the length-flag for every delta key is still redundant. Therefore, I propose that a technique which makes all delta keys into sequential partitions, where within a partition, the length for member delta keys are identical. Assigning delta keys to partitions with optimized space is possible with dynamic programming.

I denote the i-th delta key as d_i, and the maximal possible length of a partition as 2^l. Within a partition, there is a length-flag F of 2^f-bits denoting the length for every delta key is F. Denoting the minimal space required to hold first-i delta keys as $g(i)$:

$$g(i) = \min_{j=\min(1,i-2^l)}^{i-1} (g(i-j)+l+f+j\times \max_{k\in[i-j+1,i]} (\lceil \log_2 d_i \rceil))$$

This dynamic programming can be solved in linear time.

V. Conclusion

In this reading report, I analyzed the major improvement of FastSGD compared to SketchML, calculated theoretic performance metrics for specific and general scenarios, and finally I propose my ideas about further improvement and development of FastSGD.

References

[1] Keyu Yang, Lu Chen, Zhihao Zeng, and Yunjun Gao. Fastsgd: A fast compressed sgd framework for distributed machine learning, 2021.