PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization

Thijs Vogels Sai Praneeth Karimireddy Martin Jaggi

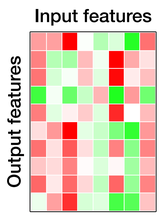
Lossy Gradient Compression

In distributed training, workers typically exchange their mini-batch gradients at every iteration. These gradients can be 100's of megabytes large, so this communication limits the scalability of distributed optimization.

Lossy compression of gradients before sharing them across workers is a popular approach to mitigate this problem.

Rapid Low-rank Approximation

PowerSGD sees a layer's gradient as a matrix. It approximates this matrix as the product of two narrow matrices by using one step of power iteration.

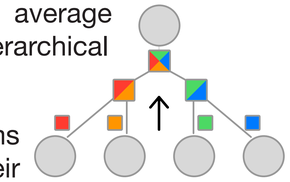


This approximation is coarse, but only involves two multiplications of the gradient matrix and a very narrow one, followed by an orthogonalization of the output. This is much faster than an SVD.

PowerSGD converges, even with this coarse approximation. This is mainly due to the error feedback mechanism.

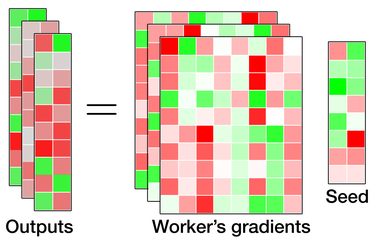
All-reduce Communication

In normal, uncompressed, SGD, the workers average their gradients after each iteration. This average can be computed efficiently with hierarchical all-reduce communication.



Unfortunately, compressed algorithms cannot hierarchically aggregate their compressed gradients. Therefore, these algorithms resort to less scalable all-to-all communication or a parameter server.

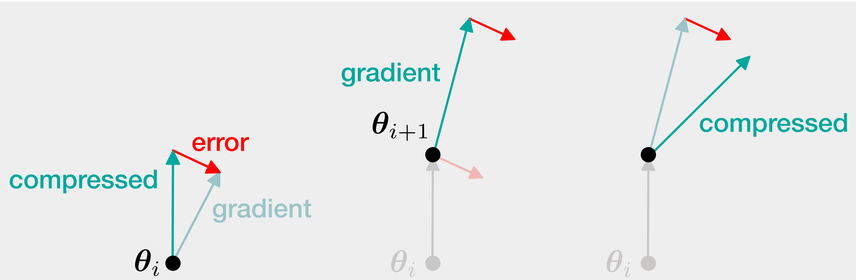
The power iteration step of PowerSGD, effectively multiplies the average gradient matrix across workers with the same narrow matrix (right). Due to linearity, this operation is equivalent to averaging the small output matrices (left).



Because all communication in PowerSGD is just an average operation, it enjoys all the benefits of all-reduce.

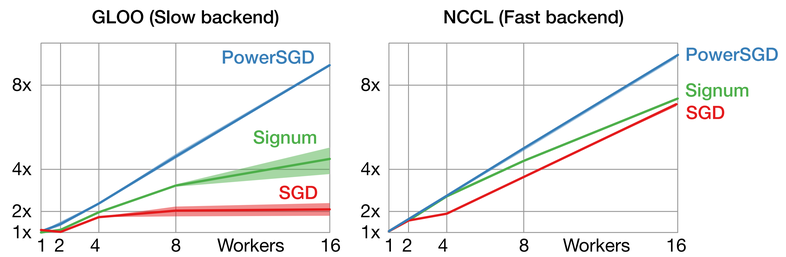
Error Feedback

Even though PowerSGD compression is *biased* and of *low quality*, the algorithm can converge in a similar number of steps as full-precision SGD. This is thanks to *error feedback* [Seide et al. 2014, Stich et al. 2018, Karimireddy et al. 2019].



Scalability

Due to its fast compression algorithm and strong reduction in communication (around 100x in our experiments), PowerSGD scales well on slow backends, but can still improve over SGD when using Nvidia's highly optimized NCCL.



ResNet-18 on Cifar-10

Plug & Play

In our experiments, PowerSGD can be used plug-and-play with an existing optimizer without re-tuning the optimizer's hyperparameters. With a high enough compression rank, PowerSGD can achieve the same test accuracy as uncompressed, full-precision SGD while enjoying reductions in communication of more than 100x.

Language modeling — LSTM on WIKITEXT-2

Algorithm	Test perplexity	Data sent per epoch		Time per batch	
SGD	91	7730 MB	(1×)	300 ms	+0%
Rank 1	102	25 MB	(310×)	131 ms	−56%
Rank 2	93	38 MB	(203×)	141 ms	−53%
Rank 4	91	64 MB	(120×)	134 ms	−55%

Code

Download the code at github.com/epfml/powersgd.