Talk 6: Compression in Federated Learning

WEN Hao

2021-7-15

Compression in Federated Learning

As is discussed previously (e.g. in the study of "GADMM" to "CQ-GGADMM"), one of the main bottleneck communication cost can be reduced using

- compression
- lazy aggregation (censoring)
- etc.

Compression in Federated Learning

As is discussed previously (e.g. in the study of "GADMM" to "CQ-GGADMM"), one of the main bottleneck communication cost can be reduced using

- compression
- lazy aggregation (censoring)
- etc.

The technique of compression mainly consists of

- (randomized) quantization
- sparsification

or their combination.

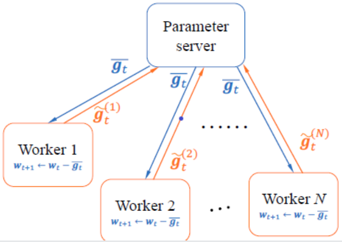
Deterministic Compression

compression can be naively done via fixed reduction of precision (fixed bit of quantization) of parameters and/or gradients, e.g. half precision (`float32` $\rightarrow$ `float16`) or mixed precision.

This is the common practice for acceleration of ordinary (non-distributed) model training process. e.g. the PyTorch Post on mixed precision training.

TernGrad

One extreme case of compression is to take the sign of each coordinate of the stochastic gradient vector, which makes it binary (1-bit, ± 1) or ternary ($\{-1, 0, +1\}$).



$\widetilde{g}_t^{(i)}$ is the ternarized gradient

$$g_t^{(i)} = \|g_t^{(i)}\|_\infty \cdot \mathrm{sign}(g_t^{(i)}) \odot \boxed{b_t}$$

where b_t is a random binary vector satisfying some Bernoulli distribution $Be(|g_{t,k}^{(i)}|/s_t)$

Similar algorithms include 1-bit SGD [1], signSGD [2]

[1] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-Bit Stochastic Gradient Descent and Application to Data-Parallel Distributed Training of Speech DNNs," in *Interspeech 2014*, 9 2014

[2] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed Optimisation for Non-Convex Problems," in *International Conference on Machine Learning*, pp. 560–569, PMLR, 2018

More generally, in QSGD [3], randomized quantization (called "low-precision quantizer" in [4]) is performed on gradients v via

$$Q_s(v) = \|v\|_2 \cdot \operatorname{sign}(v) \odot \boxed{\xi(v,s)},$$

where the i-th element in vector $\xi(v,s)$ is defined by

$$\xi_i(v,s) = \begin{cases} (\ell+1)/s, & \text{with prob. } (|v_i|/\|v\|_2)s - \ell \\ \ell/s, & \text{otherwise} \end{cases}$$

s controls the number of quantization levels, and ℓ (should be ℓ_i?) be s.t. $|v_i|/\|v\|_2 \in [\ell/s, (\ell+1)/s]$.

---

[3] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017

[4] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, "Distributed Learning with Compressed Gradients," *arXiv preprint arXiv:1806.06573*, 2018

DCGD

DCGD [4] generalized such operators Q_s into an abstract concept

Definition (Unbiased Random Quantizer (URQ))

A mapping $Q : \mathbb{R}^d \to \mathbb{R}^d$ is called an unbiased random quantizer if $\forall v \in \mathbb{R}^d$,

- $\mathrm{supp}(Q(v)) \subseteq \mathrm{supp}(v)$
- $\mathbb{E}[Q(v)] = v$
- $\mathbb{E}[\|Q(v)\|_2^2] \leqslant \alpha\|v\|_2^2$ *for some finite positive* α

And perhaps with more useful properties like

- sparsity: $\mathbb{E}[\|Q(v)\|_0] \leqslant \mathrm{const}$
- sign preserving: $Q(v)_i \cdot v_i \geqslant 0$

[4] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, "Distributed Learning with Compressed Gradients," *arXiv preprint arXiv:1806.06573*, 2018

Examples of URQs

Despite the ternary quantizer and low-precision quantizer, one has [5]

Random-k sparsification

$$\mathcal{C}(v) = \frac{d}{k}(v \odot \xi_k)$$

where $\xi_k \in \{0, 1\}^d$ is a uniformly random binary vector with k nonzero entries, $v \in \mathbb{R}^d$.

[5] Z. Li, D. Kovalev, X. Qian, and P. Richtarik, "Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 5895–5904, PMLR, 7 2020

Examples of URQs

Despite the ternary quantizer and low-precision quantizer, one has [5]

(p, s)-quantization

$$\mathcal{C}_{p,s}(v) = \operatorname{sign}(v) \cdot \|v\|_p \cdot \frac{1}{s}\xi(v, s)$$

where $\xi(v, s)$ is a random vector with i-th element

$$\xi_i(v, s) = \begin{cases} \ell_i + 1, & \text{with prob. } (|v_i|/\|v\|_2)s - \ell_i \\ \ell_i, & \text{otherwise} \end{cases}$$

and ℓ_i be s.t. $|v_i|/\|v\|_2 \in [\ell_i/s, (\ell_i + 1)/s]$

[5] Z. Li, D. Kovalev, X. Qian, and P. Richtarik, "Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 5895–5904, PMLR, 7 2020

Implementations of Quantizers

One can refer to https://github.com/burlachenkok/marina for code and examples of various compressors, e.g. in files

- linear_model_with_non_convex_loss/compressors.py
- neural_nets_experiments/compressors.py

or this simple jupyter notebook

1 Naive Compression Methods

2 Recent Development

(A)DIANA

The main contribution of (A)DIANA [5, 6] is that, instead of quantizing the gradients, the **difference of gradient updates**, i.e. instead of

$$\tilde{g}_t^{(i)} = Q(g_t^{(i)}) = Q(\nabla f_i(x_t))$$

one performs

$$\begin{cases} \tilde{g}_t^{(i)} = h_t^{(i)} + Q(\nabla f_i(x_t) - h_t^{(i)}) \\ h_{t+1}^{(i)} = h_t^{(i)} + \alpha Q(\nabla f_i(x_t) - h_t^{(i)}) \end{cases}$$

$h^{(i)}$ are "memory" maintained locally, whose average is maintained in the central server.

---

[5] Z. Li, D. Kovalev, X. Qian, and P. Richtarik, "Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 5895–5904, PMLR, 7 2020

[6] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik, "Distributed Learning with Compressed Gradient Differences," *arXiv preprint arXiv:1901.09269*, 2019

(A)DIANA

Another key point (feature) of (A)DIANA is the combination with acceleration (and variance reduction):

Algorithm 1 DIANA (n nodes)

input learning rates $\alpha > 0$ and $\{\gamma^k\}_{k\ge 0}$, initial vectors $x^0, h_1^0, \dots, h_n^0 \in \mathbb{R}^d$ and $h^0 = \frac{1}{n}\sum_{i=1}^n h_i^0$, quantization parameter $p \ge 1$, sizes of blocks $\{d_l\}_{l=1}^m$, momentum parameter $0 \le \beta < 1$

$v^0 = \nabla f(x^0)$
for $k = 0, 1, \dots$ **do**
  Broadcast x^k to all workers
  for $i = 1, \dots, n$ in parallel **do**
    Sample g_i^k such that $\mathbf{E}[g_i^k \mid x^k] = \nabla f_i(x^k)$ and let $\Delta_i^k = g_i^k - h_i^k$
    Sample $\hat{\Delta}_i^k \sim \mathrm{Quant}_p(\Delta_i^k, \{d_l\}_{l=1}^m)$ and let $h_i^{k+1} = h_i^k + \alpha\hat{\Delta}_i^k$ and $\hat{g}_i^k = h_i^k + \hat{\Delta}_i^k$
  end for
  $\hat{\Delta}^k = \frac{1}{n}\sum_{i=1}^n \hat{\Delta}_i^k; \quad \hat{g}^k = \frac{1}{n}\sum_{i=1}^n \hat{g}_i^k = h^k + \hat{\Delta}^k; \quad v^k = \beta v^{k-1} + \hat{g}^k$
  $x^{k+1} = \mathrm{prox}_{\gamma^k R}\left(x^k - \gamma^k v^k\right); \quad h^{k+1} = \frac{1}{n}\sum_{i=1}^n h_i^{k+1} = h^k + \alpha\hat{\Delta}^k$
end for

Note the "Quant" operator is a so-called "block-quantizer" or "bucket-quantizer"[3] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017

(A)DIANA

Another key point (feature) of (A)DIANA is the combination with acceleration (and variance reduction):

Algorithm 2 Accelerated DIANA (ADIANA)

Input: initial point x^0, $\{h_i^0\}_{i=1}^n$, $h^0 = \frac{1}{n}\sum_{i=1}^n h_i^0$, parameters $\eta, \theta_1, \theta_2, \alpha, \beta, \gamma, p$

$z^0 = y^0 = w^0 = x^0$

for $k = 0, 1, 2, \ldots$ **do**

  $x^k = \theta_1 z^k + \theta_2 w^k + (1 - \theta_1 - \theta_2) y^k$

  for all machines $i = 1, 2, \ldots, n$ **do in parallel**

    Compress shifted local gradient $\mathcal{C}_i^k(\nabla f_i(x^k) - h_i^k)$ and send to the server

    Update local shift $h_i^{k+1} = h_i^k + \alpha \mathcal{C}_i^k(\nabla f_i(w^k) - h_i^k)$

  end for

  Aggregate received compressed gradient information

$$g^k = \frac{1}{n}\sum_{i=1}^n \mathcal{C}_i^k(\nabla f_i(x^k) - h_i^k) + h^k$$

$$h^{k+1} = h^k + \alpha \frac{1}{n}\sum_{i=1}^n \mathcal{C}_i^k(\nabla f_i(w^k) - h_i^k)$$

  Perform update step

$$y^{k+1} = \mathrm{prox}_{\eta\psi}(x^k - \eta g^k)$$

  $z^{k+1} = \beta z^k + (1-\beta) x^k + \frac{\gamma}{\eta}(y^{k+1} - x^k)$

$$w^{k+1} = \begin{cases} y^k, & \text{with probability } p \\ w^k, & \text{with probability } 1 - p \end{cases}$$

end for

MARINA

MARINA [7] replaced the unbiased compressor by a biased one, via replacing

$$\begin{cases} \tilde{g}_t^{(i)} = h_t^{(i)} + Q(\nabla f_i(x_t) - h_t^{(i)}) \\ h_{t+1}^{(i)} = h_t^{(i)} + \alpha Q(\nabla f_i(x_t) - h_t^{(i)}) \end{cases}$$

by

$$\tilde{g}_t^{(i)} = \begin{cases} \nabla f_i(x_t), & \text{with prob. } p \\ \tilde{g}_{t-1}^{(i)} + Q(\nabla f_i(x_t) - \nabla f_i(x_{t-1})), & \text{with prob. } 1-p \end{cases}$$

for some small p.

MARINA

MARINA [7] replaced the unbiased compressor by a biased one, via replacing

$$\begin{cases} \tilde{g}_t^{(i)} = h_t^{(i)} + Q(\nabla f_i(x_t) - h_t^{(i)}) \\ h_{t+1}^{(i)} = h_t^{(i)} + \alpha Q(\nabla f_i(x_t) - h_t^{(i)}) \end{cases}$$

by

$$\tilde{g}_t^{(i)} = \begin{cases} \nabla f_i(x_t), & \text{with prob. } p \\ \tilde{g}_{t-1}^{(i)} + Q(\nabla f_i(x_t) - \nabla f_i(x_{t-1})), & \text{with prob. } 1-p \end{cases}$$

for some small p.

As claimed by the authors, their intuition come from the rare (?) phenomenon in stochastic optimization that

> *"the bias of the stochastic gradient helps to achieve better complexity"*

MARINA

The basic MARINA algorithm is as follows:

Algorithm 1 MARINA

Input: starting point x^0, stepsize γ, probability $p \in (0, 1]$, number of iterations K
Initialize $g^0 = \nabla f(x^0)$
for $k = 0, 1, \ldots, K-1$ **do**
  Sample $c_k \sim \text{Be}(p)$
  Broadcast g^k to all workers
  for $i = 1, \ldots, n$ in parallel **do**
    $x^{k+1} = x^k - \gamma g^k$
    Set $g_i^{k+1} = \nabla f_i(x^{k+1})$ if $c_k = 1$, and $g_i^{k+1} = g^k + \mathcal{Q}\left(\nabla f_i(x^{k+1}) - \nabla f_i(x^k)\right)$ otherwise
  end for
  $g^{k+1} = \frac{1}{n}\sum_{i=1}^{n} g_i^{k+1}$
end for
Return: $\hat{x}^K$ chosen uniformly at random from $\{x^k\}_{k=0}^{K-1}$

More?

- higher order methods [8, 9]
- combination with lazy aggregation [10], and with stochastic update
- biased compression [11, 12]
- analysis of communication cost (# rounds and bandwidth)

More?

- higher order methods [8, 9]
- combination with lazy aggregation [10], and with stochastic update
- biased compression [11, 12]
- analysis of communication cost (# rounds and bandwidth)

and more, to be continued...

Additional resources from FLOW

- MARINA: Faster Non-Convex Distributed Learning with Compression
- On Biased Compression for Distributed Learning

References I

[1] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-Bit Stochastic Gradient Descent and Application to Data-Parallel Distributed Training of Speech DNNs," in *Interspeech 2014*, 9 2014.

[2] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed Optimisation for Non-Convex Problems," in *International Conference on Machine Learning*, pp. 560–569, PMLR, 2018.

[3] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017.

[4] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, "Distributed Learning with Compressed Gradients," *arXiv preprint arXiv:1806.06573*, 2018.

References II

[5] Z. Li, D. Kovalev, X. Qian, and P. Richtarik, "Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 5895–5904, PMLR, 7 2020.

[6] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik, "Distributed Learning with Compressed Gradient Differences," *arXiv preprint arXiv:1901.09269*, 2019.

[7] E. Gorbunov, K. Burlachenko, Z. Li, and P. Richtárik, "MARINA: Faster Non-Convex Distributed Learning with Compression," *arXiv preprint arXiv:2102.07845*, 2021.

[8] R. Crane and F. Roosta, "DINGO: Distributed Newton-Type Method for Gradient-Norm Optimization," *Advances in Neural Information Processing Systems 32 (Nips 2019)*, vol. 32, 2019.

References III

[9] R. Islamov, X. Qian, and P. Richtárik, "Distributed Second Order Methods with Fast Rates and Compressed Communication," *arXiv preprint arXiv:2102.07158*, 2021.

[10] C. B. Issaid, A. Elgabli, J. Park, and M. Bennis, "Communication Efficient Distributed Learning with Censored, Quantized, and Generalized Group ADMM," *arXiv preprint arXiv:2009.06459*, 2020.

[11] A. Beznosikov, S. Horváth, P. Richtárik, and M. Safaryan, "On Biased Compression for Distributed Learning," *arXiv preprint arXiv:2002.12410*, 2020.

[12] M. Safaryan, E. Shulgin, and P. Richtárik, "Uncertainty Principle for Communication Compression in Distributed and Federated Learning and the Search for an Optimal Compressor," *arXiv preprint arXiv:2002.08958*, 2020.