Quantization in Federated Learning

Tao Shen

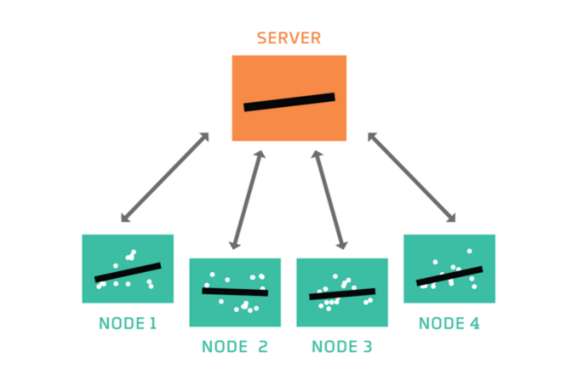
Zhejiang University

November 23, 2019

FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization

Federated Learning

Federated Learning



Communication bottleneck

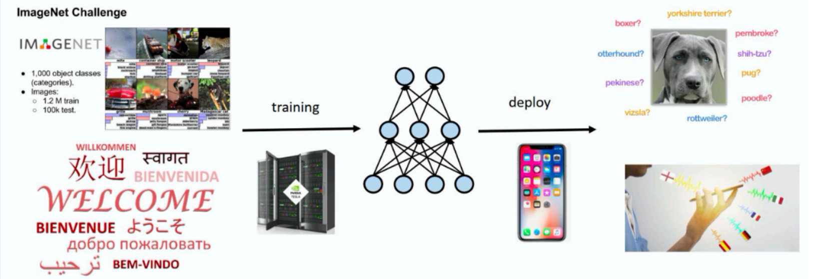
Communication-efficient? Large number of parameters

Model Compress

Small model

better generalization, faster prediction, smaller memory footprint

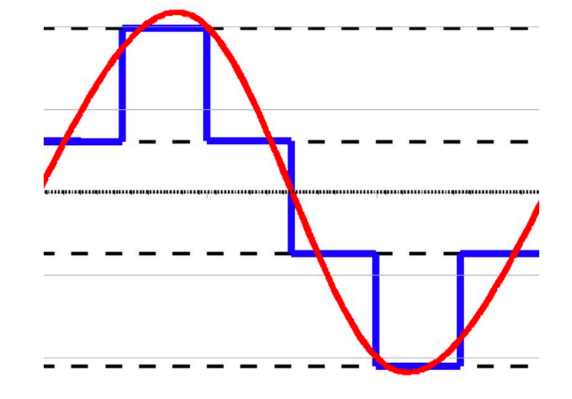
Computation and memory intensive



capacity of deep network is usually larger than needed

Quantization

Quantization



Weight Quantization

32-bit $\rightarrow$ fewer bits

FedPAQ

Periodic Update

...

Partial participation

...

Weight quantization

!!! how?

Quantizer

Quantizer in FedPAQ

Example 1 (Low-precision quantizer [Alistarh et al., 2017]). *For any variable* $\mathbf{x} = [x_1, \cdots, x_p]^\top$*, the low precision quantizer* $Q^{LP} : \mathbb{R}^p \to \mathbb{R}^p$ *is defined as below*

$$Q^{LP}(\mathbf{x}) = [Q_1^{LP}(\mathbf{x}), \cdots, Q_p^{LP}(\mathbf{x})]^\top, \quad \text{and} \quad Q_i^{LP}(\mathbf{x}) = \|\mathbf{x}\| \cdot \mathrm{sign}(x_i) \cdot \xi_i(\mathbf{x}, s), \tag{5}$$

where $\xi_i(\mathbf{x}, s)$ *is a random variable defined as*

$$\xi_i(\mathbf{x}, s) = \begin{cases} \frac{l}{s} & w.p. \;\; 1 - \left(\frac{|x_i|}{\|\mathbf{x}\|} s - l\right), \\ \frac{l+1}{s} & w.p. \;\; \frac{|x_i|}{\|\mathbf{x}\|} s - l. \end{cases} \tag{6}$$

In above, the tuning parameter s *corresponds to the number of quantization levels and* $l \in [0, s)$ *is an integer such that* $|x_i|/\|\mathbf{x}\| \in [l/s, (l+1)/s)$.

Theoretical guarantee

FedPAQ

Algorithm 1 FedPAQ

Require: number of periods K, period length τ, stepsize $\eta_{k,t}$

for $k = 0, 1, \cdots, K-1$ **do**

  parameter server picks r nodes uniformly at random denoted by $\mathcal{S}_k \subseteq [n]$

  parameter server sends $\mathbf{x}_k$ to all the nodes in $\mathcal{S}_k$

  for node $i \in \mathcal{S}_k$ **do**

    $\mathbf{x}_{k,0}^{(i)} \leftarrow \mathbf{x}_k$

    for $t = 0, 1, \cdots, \tau - 1$ **do**

      randomly pick a data point $\xi \in \mathcal{D}^i$ and

      compute the stochastic gradient $\widetilde{\nabla} f_i(\mathbf{x}) = \nabla \ell(\mathbf{x}, \xi)$

      $\mathbf{x}_{k,t+1}^{(i)} \leftarrow \mathbf{x}_{k,t}^{(i)} - \eta_{k,t} \widetilde{\nabla} f_i \left(\mathbf{x}_{k,t}^{(i)} \right)$

    end for

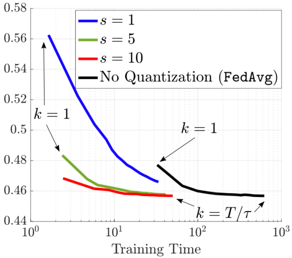
    send $Q\left(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k \right)$ to the parameter server

  end for

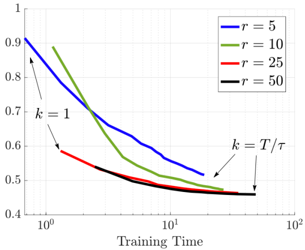
  parameter server updates $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \frac{1}{r} \sum_{i \in \mathcal{S}_k} Q\left(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k \right)$

end for

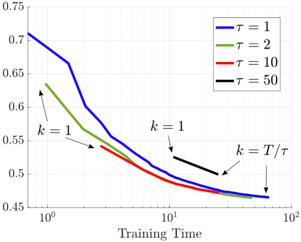
FedPAQ



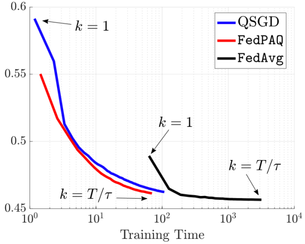
(a) varying the quantization level s



(b) varying the number of active nodes r



(c) varying the period length τ



(d) comparing **FedPAQ**, **FedAvg**, and QSGD

Loss-aware Binarization of Deep Networks

Loss-aware Binarization

Loss-aware

explicitly consider the loss during quantization

Binarization

$\{+1,-1\}$

Loss-aware Binarization

Loss-aware Binarization

$$\begin{array}{ll} \min_{\hat{\mathbf{w}}} & \ell(\hat{\mathbf{w}}) \\ \text{s.t.} & \hat{\mathbf{w}}_l = \alpha_l \mathbf{b}_l, \alpha_l > 0, \mathbf{b}_l \in \{\pm 1\}^{n_l}, l = 1, \ldots, L \end{array}$$

proximal Newton algorithm

$$\ell\left(\hat{\mathbf{w}}^{t-1}\right) + \nabla \ell\left(\hat{\mathbf{w}}^{t-1}\right)^\top \left(\hat{\mathbf{w}}^{t} - \hat{\mathbf{w}}^{t-1}\right) + \frac{1}{2}\left(\hat{\mathbf{w}}^{t} - \hat{\mathbf{w}}^{t-1}\right)^\top \mathbf{H}^{t-1}\left(\hat{\mathbf{w}}^{t} - \hat{\mathbf{w}}^{t-1}\right)$$

Approximate diagonal Hessian

$$\begin{array}{ll} \min_{\hat{\mathbf{w}}^t} & \nabla \ell\left(\hat{\mathbf{w}}^{t-1}\right)^\top \left(\hat{\mathbf{w}}^{t} - \hat{\mathbf{w}}^{t-1}\right) + \frac{1}{2}\left(\hat{\mathbf{w}}^{t} - \hat{\mathbf{w}}^{t-1}\right)^\top \mathbf{D}^{t-1}\left(\hat{\mathbf{w}}^{t} - \hat{\mathbf{w}}^{t-1}\right) \\ \text{s.t.} & \hat{\mathbf{w}}_l^t = \alpha_l^t \mathbf{b}_l^t, \alpha_l^t > 0, \mathbf{b}_l^t \in \{\pm 1\}^{n_l}, \quad l = 1, \ldots, L \end{array}$$

Loss-aware Binarization

Closed-form optimal solution

Proposition 3.1 *Let* $\mathbf{d}_l^{t-1} \equiv diag(\mathbf{D}_l^{t-1})$, *and*

$$\mathbf{w}_l^t \equiv \hat{\mathbf{w}}_l^{t-1} - \nabla_l \ell(\hat{\mathbf{w}}^{t-1}) \oslash \mathbf{d}_l^{t-1}. \tag{7}$$

The optimal solution of (6) can be obtained in closed-form as

$$\alpha_l^t = \frac{\|\mathbf{d}_l^{t-1} \odot \mathbf{w}_l^t\|_1}{\|\mathbf{d}_l^{t-1}\|_1}, \quad \mathbf{b}_l^t = sign(\mathbf{w}_l^t). \tag{8}$$

loss-aware quantized weights can be easily computed

Loss-aware Binarization

Algorithm 1 Loss-Aware Binarization (LAB) for training a feedforward neural network.

Input: Minibatch $\{(\mathbf{x}_0^t, \mathbf{y}^t)\}$, current full-precision weights $\{\mathbf{w}_l^t\}$, first moment $\{\mathbf{m}_l^{t-1}\}$, second moment $\{\mathbf{v}_l^{t-1}\}$, and learning rate η^t.

```
Forward Propagation
for l = 1 to L do
    α_l^t = ‖d_l^{t-1} ⊙ w_l^t‖_1 / ‖d_l^{t-1}‖_1;
    b_l^t = sign(w_l^t);
    rescale the layer-l input: x̃_{l-1}^t = α_l^t x_{l-1}^t;
    compute z_l^t with input x̃_{l-1}^t and binary weight b_l^t;
    apply batch-normalization and nonlinear activation to z_l^t to obtain x_l^t;
end for
compute the loss ℓ using x_L^t and y^t;
Backward Propagation
initialize output layer's activation's gradient ∂ℓ/∂x_L^t;
for l = L to 2 do
    compute ∂ℓ/∂x_{l-1}^t using ∂ℓ/∂x_l^t, α_l^t and b_l^t;
end for
Update parameters using Adam
for l = 1 to L do
    compute gradients ∇_l ℓ(ŵ^t) using ∂ℓ/∂x_l^t and x_{l-1}^t;
    update first moment m_l^t = β_1 m_l^{t-1} + (1 − β_1) ∇_l ℓ(ŵ^t);
    update second moment v_l^t = β_2 v_l^{t-1} + (1 − β_2)(∇_l ℓ(ŵ^t) ⊙ ∇_l ℓ(ŵ^t));
    compute unbiased first moment m̂_l^t = m_l^t / (1 − β_1^t);
    compute unbiased second moment v̂_l^t = v_l^t / (1 − β_2^t);
    compute current curvature matrix d_l^t = (1/η^t)(ε1 + √(v̂_l^t));
    update full-precision weights w_l^{t+1} = w_l^t − m̂_l^t ⊘ d_l^t;
    update learning rate η^{t+1} = UpdateRule(η^t, t + 1);
end for
```

Loss-aware Binarization

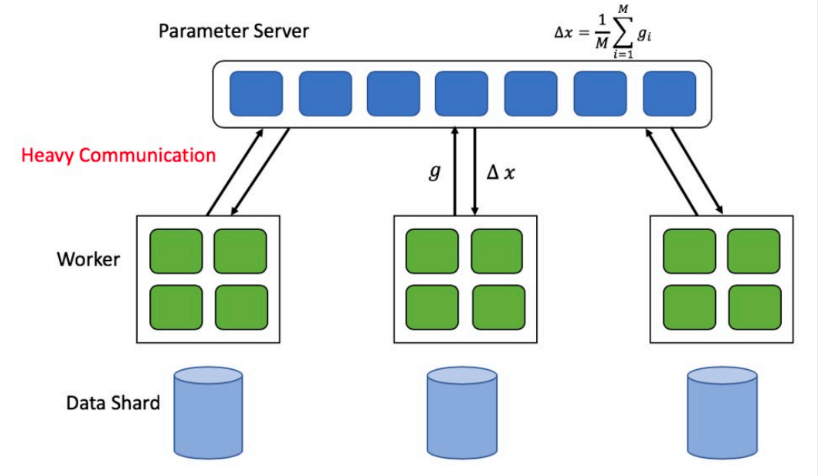
Table 1: Test error rates (%) for feedforward neural network models.

		MNIST	*CIFAR-10*	*SVHN*
(no binarization)	full-precision	1.190	11.900	2.277
(binarize weights)	BinaryConnect	1.280	**9.860**	2.450
	BWN	1.310	10.510	2.535
	LAB	**1.180**	10.500	**2.354**
(binarize weights and activations)	BNN	1.470	12.870	3.500
	XNOR	1.530	12.620	3.435
	LAB2	**1.380**	**12.280**	**3.362**

Analysis of Quantized Models

Gradient Quantization?

Distributed Machine Learning



quantize gradients to m bits

Weight or Gradient?

Weight or Gradient?

1. full-precision gradients and quantized weights (li et al., 2017, de et al., 2018);
2. full-precision weights and quantized gradients (alistarh et al., 2017, wen et al., 2017, bernstein et al., 2018);

Both Weight & Gradient

Awosome!

PRELIMINARIES

Loss-aware Weight Quantization

$$\begin{aligned} \min_{\hat{\mathbf{w}}} \quad & \|\mathbf{w}_{t+1} - \hat{\mathbf{w}}\|^2_{\mathrm{Diag}(\sqrt{\gamma_t})} \\ \text{s.t.} \quad & \hat{\mathbf{w}} = \alpha \mathbf{b}, \alpha > 0, \mathbf{b} \in (\mathcal{S}_w)^d \end{aligned}$$

Stochastic Gradient Quantization

Example 1 (Low-precision quantizer [Alistarh et al., 2017]). *For any variable* $\mathbf{x} = [x_1, \cdots, x_p]^\top$*, the low precision quantizer* $Q^{LP} : \mathbb{R}^p \to \mathbb{R}^p$ *is defined as below*

$$Q^{LP}(\mathbf{x}) = [Q_1^{LP}(\mathbf{x}), \cdots, Q_p^{LP}(\mathbf{x})]^\top, \quad \text{and} \quad Q_i^{LP}(\mathbf{x}) = \|\mathbf{x}\| \cdot \mathrm{sign}(x_i) \cdot \xi_i(\mathbf{x}, s), \tag{5}$$

where $\xi_i(\mathbf{x}, s)$ *is a random variable defined as*

$$\xi_i(\mathbf{x}, s) = \begin{cases} \frac{l}{s} & w.p. \;\; 1 - \left(\frac{|x_i|}{\|\mathbf{x}\|} s - l\right), \\ \frac{l+1}{s} & w.p. \;\; \frac{|x_i|}{\|\mathbf{x}\|} s - l. \end{cases} \tag{6}$$

In above, the tuning parameter s *corresponds to the number of quantization levels and* $l \in [0, s)$ *is an integer such that* $|x_i|/\|\mathbf{x}\| \in [l/s, (l+1)/s)$.

Weight Quantization with full-precision gradients

Update

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \operatorname{Diag}(\sqrt{\hat{\mathbf{v}}_t})^{-1}\hat{\mathbf{g}}_t$$

Speed

$$O(1/\sqrt{T})$$

Error

$$LD\sqrt{D^2 + \frac{d\alpha^2\Delta_w^2}{4}}$$

Weight Quantization with quantized gradients

Update

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \, \mathrm{Diag}(\sqrt{\tilde{\mathbf{v}}_t})^{-1} \tilde{\mathbf{g}}_t$$

Speed

slow down by a factor

$$\sqrt{\frac{1+\sqrt{2d-1}}{2}\Delta_g + 1}$$

Error

No change

Problems

1. deep networks typically have a large d;
2. distributed learning prefers a large Δ_g

Weight Quantization with clipped quantized gradients

Update

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \, \mathrm{Diag}(\sqrt{\mathbf{v}_t})^{-1} \check{\mathbf{g}}_t$$

Speed

slow down by a factor

$$\sqrt{(2/\pi)^{\frac{1}{2}} c \Delta_g + 1}$$

Error

introduce extra error

$$\sqrt{d} D \sigma (2/\pi)^{\frac{1}{4}} \sqrt{F(c)}$$

Weight Quantization with clipped quantized gradients

Speed & Error

slow down by a factor

$$\sqrt{(2/\pi)^{\frac{1}{2}} c\Delta_g + 1}$$

introduce extra error

$$\sqrt{d} D\sigma(2/\pi)^{\frac{1}{4}}\sqrt{F(c)}$$

A larger c leads to

smaller F(c), and thus smaller error
slower convergence

Test

weight / gradient	FP	LAB	LAQ2	LAQ3	LAQ4
FP	83.74	80.37	82.11	83.14	83.35
SQ2 (no clipping)	81.40	78.67	80.27	81.27	81.38
SQ2 (clip,$c = 3$)	82.99	80.25	81.59	83.14	83.40
SQ3 (no clipping)	83.24	80.18	81.63	82.75	83.17
SQ3 (clip, $c = 3$)	83.89	80.13	81.77	82.97	83.43
SQ4 (no clipping)	83.64	80.44	81.88	83.13	83.47
SQ4 (clip, $c = 3$)	83.80	79.27	81.42	82.77	83.43

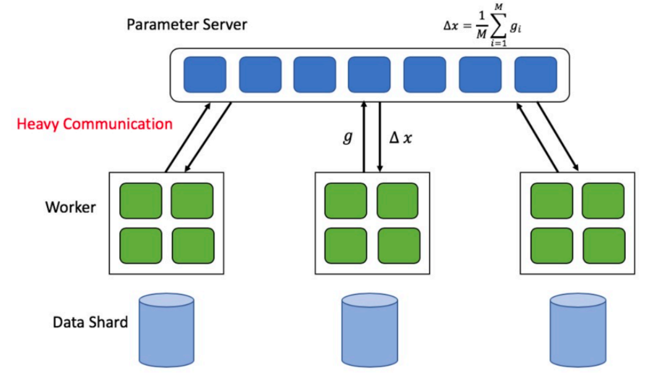
Main Findings

- An error related to the weight quantization resolution Δ_w and dimension d.
- Slow convergence by a factor related to gradient quantization resolution Δ_g and d.
- Gradient clipping renders the speed degradation mentioned above dimension-free.
- Distributed training of weight-quantized networks is much faster, while comparable accuracy with the use of full-precision gradients is maintained.

Communication-Efficient Distributed Blockwise Momentum SGD with Error-Feedback

Gradient Quantization

Distributed Machine Learning



two-way compression

same convergence rates as full-precision distributed SGD

Error-feedback

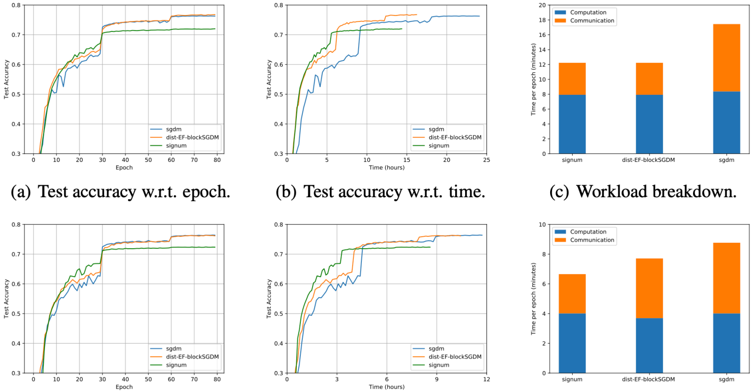
Error-feedback

Algorithm 1 SGD with Error-Feedback (EF-SGD) [11]

Input: stepsize η; compressor $\mathcal{C}(\cdot)$.
Initialize: $x_0 \in \mathbb{R}^d$; $e_0 = 0 \in \mathbb{R}^d$
for $t = 0, \dots, T-1$ **do**
  $p_t = \eta g_t + e_t$ {stochastic gradient $g_t = \nabla f(x_t, \xi_t)$}
  $\Delta_t = \mathcal{C}(p_t)$ {compressed value output}
  $x_{t+1} = x_t - \Delta_t$
  $e_{t+1} = p_t - \Delta_t$
end for

Error-feedback

Error-feedback



(a) Test accuracy w.r.t. epoch. (b) Test accuracy w.r.t. time. (c) Workload breakdown.

(d) Test accuracy w.r.t. epoch. (e) Test accuracy w.r.t. time. (f) Workload breakdown.

Figure 3: Distributed training results on the ImageNet dataset. Top: 7 workers; Bottom: 15 workers.

The End