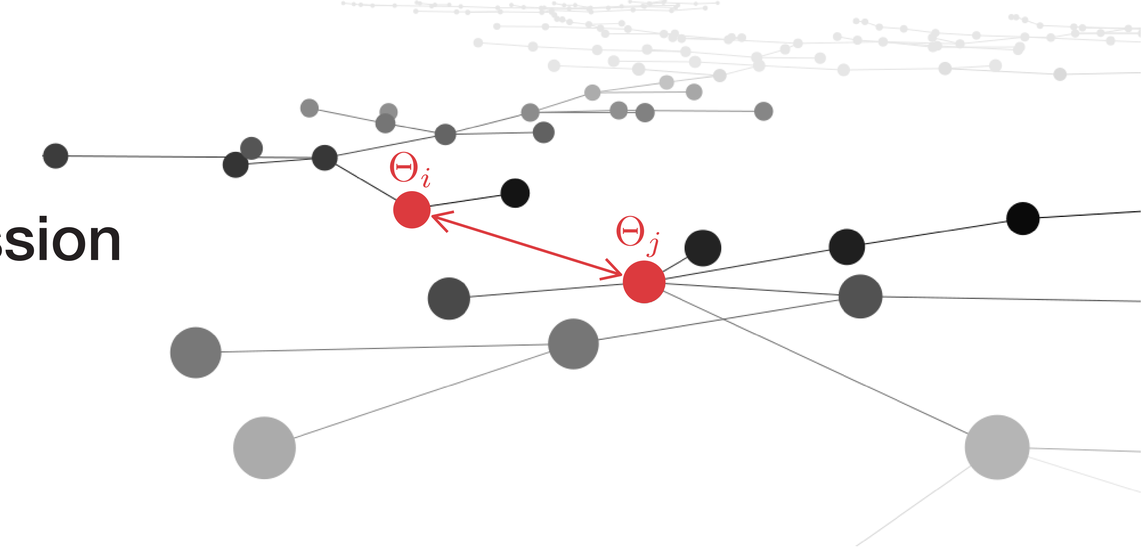
Practical Low-Rank Communication Compression in Decentralized Deep Learning

Thijs Vogels Sai Praneeth Karimireddy Martin Jaggi



PowerGossip reduces communication between workers in decentralized learning. It converges faster than related approaches and is easier to use. It matches the quality of state-of-the-art methods with less hyper-parameter tuning.

Decentralized training

- Sparsely connected **network of workers** with their own data. They communicate only with **few 'neighbors'**. Updates diffuse slowly through the network.
- The worker's parameters are **not synchronized**: everyone computes gradients using **different models**.
- A worker i does **updates** $\Theta_i \leftarrow \Theta_i - \eta \nabla f_i^{\xi}(\Theta_i)$
  and **averages** with others $\Theta_i \leftarrow \Theta_i + \sum_{j \in \mathcal{N}_i} w_{ij}(\Theta_j - \Theta_i)$ (weights based on the network)

Lossy gradient compression

- Because neural networks and their **gradients** can be gigabytes **large**, **communication** between workers can be a **performance bottleneck** in multi-worker training.
- **Lossy compression** is an attractive solution for this.
  - In the **centralized setting**, with fully-connected workers that always agree on the model parameters, this is a **practical** and **established** approach.
  - In the **decentralized setting**, methods require an additional **consensus step size** for convergence.

No additional hyper-parameters

- Contrary to related work, **PowerGossip** has **no consensus step size hyper-parameter** and works with the same learning rate as in the centralized setting.

PowerGossip

Approximating edge-wise differences

- PowerGossip **compresses parameter differences** between neighboring workers:

$$\Theta_i \leftarrow \Theta_i + \sum_{j \in \mathcal{N}_i} w_{ij} \mathcal{C}(\Theta_j - \Theta_i)$$

- Because Θ_i and Θ_j are on **different machines**, directly compressing the difference is only **possible** because PowerGossip uses a **linear** low-rank **compressor**.

Linear low-rank compression

- Based on **PowerSGD** (Vogels et al. 2019).
- The **weights** Θ of any neural network layer can be considered **matrix-shaped**. Its rows and columns correspond to the layer's inputs and outputs.
- For a normalized **vector** v, $vv^\top M$ **approximates** M.
- Workers can exchange **small vectors** instead of **matrices**:

$$vv^\top(\Theta_2 - \Theta_1) = v(v^\top\Theta_2 - v^\top\Theta_1)$$

(Just a vector; Just a vector)

Power iteration

- The **quality** of approximation depends on the vector v. The approximation is **optimal** if v is the top eigenvector.
- The output of $v^\top(\Theta_2 - \Theta_1)$ can be used as a **better-than-random** vector for the next iteration, **improving our results empirically**. These are like steps of power iteration.
- The **strength of the compression** can be controlled through the **number** of power iteration **steps** per round.

Theoretical results

- PowerGossip **consensus** has a **Q-linear rate.** The rate depends linearly on both the compression quality and on the spectral gap of the network. Related work has a **slower rate** with quadratic dependence on the **network.**
- Our convergence rate in **decentralized SGD** is asymptotically **independent** of the **compression rate** (number of averaging steps) and **network properties**.

Experimental results

- Experiments on **image classification** with a ResNet-18 and **language modeling** with an LSTM.
  - PowerGossip achieves a **similar accuracy** as alternative methods but with **no new hyper-parameters to tune**.
  - We can use the **same learning rate** as uncompressed, centralized SGD. This is possible in the same number of iterations as long as the compression is not too strong.

CIFAR-10 in a 16-worker ring topology (300 epochs):

Algorithm	η (l.r.)	γ	θ	Test accuracy	Sent/epoch
Uncompressed (DP-SGD)	tuned			92.1%	102 MB
Choco (Sign+Norm)	tuned	tuned		92.0%	3.2 MB (32×)
Moniqua (2-bit)	tuned	tuned	tuned	90.7%	6.4 MB (16×)
DeepSqueeze (Sign+Norm)	tuned	tuned		91.2%	3.2 MB (32×)
PowerGossip (1 iteration)	default			91.7%	1.8 MB (57×)
PowerGossip (2 iterations)	default			91.9%	3.0 MB (34×)

learning rate η, averaging stepsize γ

Code

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