On Communication Bottleneck in Distributed Optimization

Aleksandr Beznosikov

MIPT, HSE

30 June 2021

Content

Problem

- Distributed optimization problem:

$$\min_x f(x) := \frac{1}{M} \sum_{m=1}^{M} f_m(x). \tag{1}$$

- Machine learning interpretation – loss function:

x – weights of the ML model,

f_m – local loss depending on local data,

f – global loss.

- f_m is known only to device m – need to communicate.
- Typically, in machine learning we have access to some stochastic realisation $\nabla f_m(x, \xi)$.

Centralized approach

- Communication is done using a central server.
- All devices can send information to the server and receive responses from it.

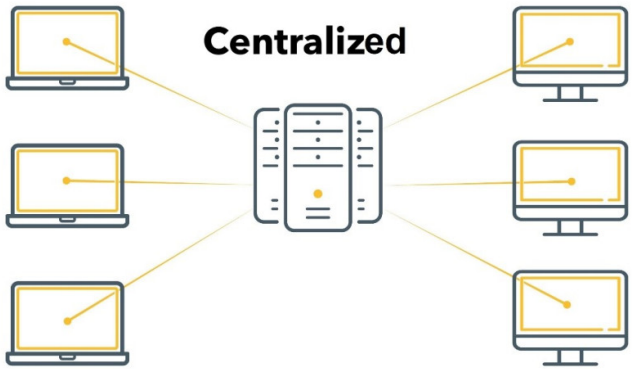

Figure: Centralized approach

- Communication takes up significantly more time than any local computation.

Centralized Stochastic Gradient Descent

Problem (1) can be solved using stochastic gradient descent. In its simplest form, this algorithm can be written as follows:

- Server sends to devices current point x^k.
- All devices compute $\nabla f_m(x^k, \xi_m^k)$.
- Devices send $\nabla f_m(x^k, \xi_m^k)$ to the server.
- Server computes $\nabla f(x^k, \xi^k) = \frac{1}{M} \sum_{m=1}^{M} f_m(x^k, \xi_m^k)$.
- Server makes a step: $x^{k+1} = x^k - \gamma \nabla f(x^k, \xi^k)$.

Centralized Stochastic Gradient Descent

We need to communicate two times per iteration.

- Server sends to devices current point x^k.
- All devices compute $\nabla f_m(x^k, \xi_m^k)$.
- Devices send $\nabla f_m(x^k, \xi_m^k)$ to the server.
- Server computes $\nabla f(x^k, \xi^k) = \frac{1}{M} \sum_{m=1}^{M} f_m(x^k, \xi_m^k)$.
- Server makes a step: $x^{k+1} = x^k - \gamma \nabla f(x^k, \xi^k)$.

Local gradient computation and gradient descent step on the server takes less time than information transfer.

Assumptions

- **Assumption 1.** $f(x)$ is Lipschitz continuous with constant L, i.e. for all x_1, x_2

$$\|\nabla f_m(x_1) - \nabla f_m(x_2)\| \leq L\|x_1 - x_2\|.$$

- **Assumption 2.** $f(x)$ is strongly-convex-strongly-concave with constant μ.
- **Assumption 3.** $\nabla f_m(x, \xi)$ is unbiased and has bounded variance, i.e. for all x

$$\mathbb{E}[\nabla f_m(x, \xi)] = \nabla f_m(x), \ \mathbb{E}[\|\nabla f_m(x, \xi) - \nabla f_m(x)\|^2] \leq \sigma^2.$$

Local and MiniBatch methods

1st idea – more local computations

- For Centralized SGD from previous slide: one communication corresponds to one computation of $\nabla f_m(x, \xi)$.
- If computing the gradient is much cheaper in terms of time, why not count it not once, but several times.
- There are two different approaches to how to implement this idea.

Local SGD

The idea of this method:

- To make local steps on each of the devices:

$$x_m^{k+1} = x_m^k - \gamma \nabla f_m(x_m^k, \xi_m^k).$$

- Every Kth iteration send current x_m^k to the server. Server averages the value $x^k = \frac{1}{M} \sum_{m=1}^{M} x_m^k$, and then sends new x^k to devices. Devices: $x_m^k = x^k$.
- A common centralized SGD is Local SGD with $K = 1$.

Convergence

- Typical convergence of this method in practice:

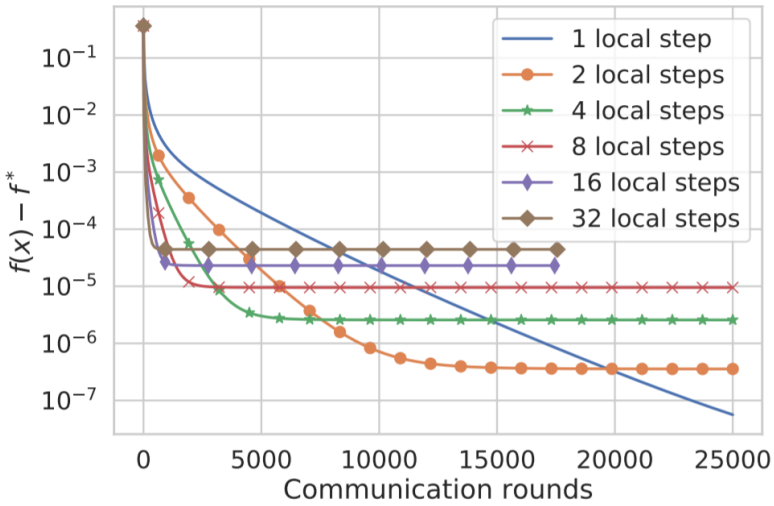

Figure: Convergence of local SGD in practice for logistic regression. Plot from here.

- Faster in terms of communications, the worse the quality of the solution.

MiniBatch SGD

The idea of this method:

- Centralized SGD but, batch size of K:

$$\nabla f(x^k, \xi_m^k) \text{ in common version}$$

$$\frac{1}{b}\sum_{i=1}^{b} \nabla f(x^k, \xi_m^{k,i}) \text{ in MiniBatch version.}$$

- A common centralized SGD is MiniBacth SGD with $b = 1$.
- If $\sigma = 0$ (non-stochastic case). If the case is non-stochastic, there is no point in collecting the batch, since the batch simply reduces the variance:

$$\sigma^2 \to \frac{\sigma^2}{b}.$$

- Convergence – similar to the convergence of ordinary SGD.

Which method is better?

- It is impossible to give an answer to this question. Local SGD is faster in terms of the number of communications, but MiniBatch is better in terms of the quality of the solution.
- Optimal strategy that can be proposed:
  1) use Local SGD, gradually decreasing K – the number of local steps,
  2) with $K = 1$, this is Centralized SGD,
  3) use MiniBatch and increase the size of the batch b to achieve the most accurate solution.
- From the point of view of theory, it is also impossible to identify the optimal method.

Literature

- Convergence theory for Local method:
  1) Parallelized Stochastic Gradient Descent (Zinkevich et al.)
  2) Local SGD Converges Fast and Communicates Little (Stich)
  3) Tighter Theory for Local SGD on Identical and Heterogeneous Data (Khaled et al.)
- The question of optimality and lower bounds for MiniBatch and Local method was investigated in a series of works:
  1) Is Local SGD Better than Minibatch SGD? (Woodworth et al.)
  2) Minibatch vs Local SGD for Heterogeneous Distributed Learning (Woodworth et al.)
  3) The Min-Max Complexity of Distributed Stochastic Convex Optimization with Intermittent Communication (Woodworth et al.)

Quantization and Compression

2d idea – compress information

- In the previous idea, we reduced the number of communications. In this idea we do not reduce the number of communications, but compress the information that we transmit.
- Let us define the quantization operators (unbiased operators):

$$\mathbb{E}Q(x) = x, \ \mathbb{E}\|Q(x) - x\|^2 = \omega x.$$

- As such operators, one can take, for example, the choice of a random number of coordinates.
- The compression operators (biased operators):

$$\mathbb{E}\|C(x) - x\|^2 = \omega x.$$

- For example, the choice of top number (in magnitude) of coordinates, sign operators, rounding operators.

Centralized Stochastic Gradient Descent with Quantization

- Server sends to devices current point x^k.
- All devices compute $\nabla f_m(x^k, \xi_m^k)$.
- Devices send $Q(\nabla f_m(x^k, \xi_m^k))$ to the server.
- Server computes $\frac{1}{M} \sum_{m=1}^{M} Q(f_m(x^k, \xi_m^k))$.
- Server makes a step: $x^{k+1} = x^k - \gamma \cdot \frac{1}{M} \sum_{m=1}^{M} Q(f_m(x^k, \xi_m^k))$.

Centralized Stochastic Gradient Descent with Compression

Compression is not so simple - you need to use the error feedback technique:

- Server sends to devices current point x^k.
- All devices compute $\nabla f_m(x^k, \xi_m^k)$.
- Devices send $C(e_i^k + \gamma \nabla f_m(x^k, \xi_m^k))$ to the server.
- All devices compute $e_i^{k+1} = e_i^k + \gamma \nabla f_m(x^k, \xi_m^k) - C(e_i^k + \gamma \nabla f_m(x^k, \xi_m^k))$.
- Server computes $\frac{1}{M} \sum_{m=1}^{M} C(e_i^k + \gamma \nabla f_m(x^k, \xi_m^k))$.
- Server makes a step: $x^{k+1} = x^k - \frac{1}{M} \sum_{m=1}^{M} C(e_i^k + \gamma \nabla f_m(x^k, \xi_m^k))$.

Here we need additional error sequence.

Convergence

- Comparison of SGD, method with compression and quantization for VGG19 training:

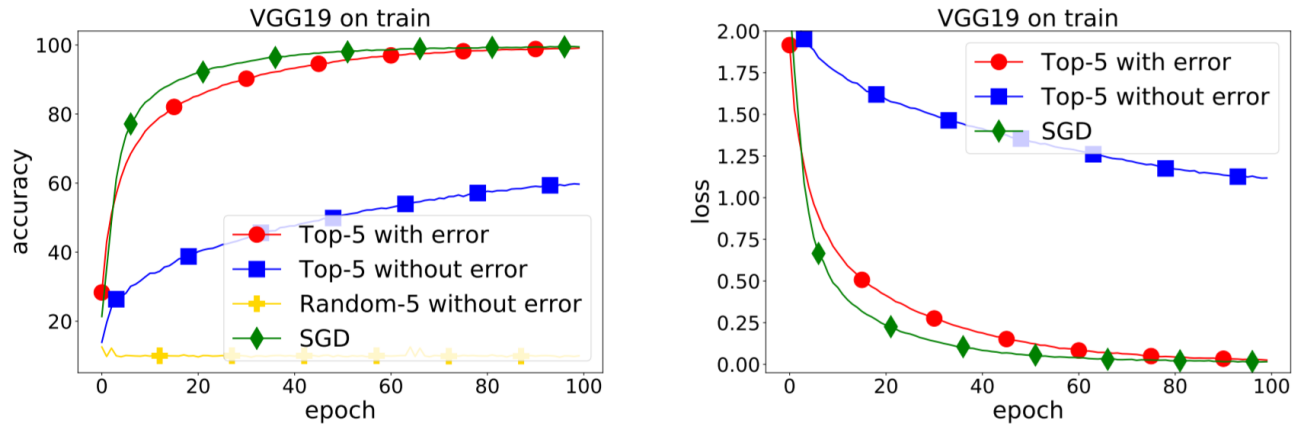

Figure: Comparison of SGD, method with compression and quantization for VGG19 training. Plot from here.

- For the number of epochs, the method with compression converges almost the same as the method without compression, but the method with compression transmits much less information.

Literature

- About quantization and compression:
  1) QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding (Alistarh et al.)
  2) On Biased Compression for Distributed Learning (Beznosikov et al.)
- Error feedback methods:
  1) Sparsified SGD with Memory (Stich et al.)
  2) Linearly Converging Error Compensated SGD (Gorbunov et al.)
  3) EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback (Richtarick et al.)

Data similarity

3d idea – use statistical similarity

- Divide data evenly across all devices. Each device has n data points.
- Then with probability $1 - \delta$, it holds

$$\left\|\nabla^2 f - \nabla^2 f_m\right\|^2 \leq \sqrt{\frac{32L^2 \log(d/\delta)}{n}} = \alpha$$

- Also, one can prove that for $\phi(x) = f_1(x) + \frac{\alpha}{2}\|x\|^2$

$$\frac{\mu}{\mu + 2\alpha}\nabla^2\phi(x) \leq \nabla^2 f(x) \leq \nabla^2\phi(x).$$

Method for data similarity

- Server sends to devices current point x^k.
- All devices compute $\nabla f_m(x^k)$.
- Devices send $\nabla f_m(x^k)$ to the server.
- Server computes $\nabla f(x^k) = \frac{1}{M} \sum\limits_{m=1}^{M} f_m(x^k)$.
- Server makes a step:

$$x^{k+1} = \arg\min_x \left(\langle \nabla f(x^k); x \rangle + \frac{1}{2\gamma} V_\phi(x, x^k) \right),$$

where Bregman divergence
$V_\phi(x, x^k) = \phi(x) - \phi(x^k) - \langle \nabla \phi(x^k); x - x^k \rangle$.

Method for data similarity

- Server sends to devices current point x^k.
- All devices compute $\nabla f_m(x^k)$.
- Devices send $\nabla f_m(x^k)$ to the server.
- Server computes $\nabla f(x^k) = \frac{1}{M} \sum\limits_{m=1}^{M} f_m(x^k)$.
- Server makes a step:

$$x^{k+1} = \arg\min_x \left(\langle \nabla f(x^k); x \rangle + \frac{1}{2\gamma} V_\phi(x, x^k) \right),$$

where Bregman divergence
$V_\phi(x, x^k) = \phi(x) - \phi(x^k) - \langle \nabla \phi(x^k); x - x^k \rangle$.

Convergences

An estimate of the number of communications that needs to be done to achieve accuracy ε:

- Centralized Gradient Descent:

$$\mathcal{O}\left(\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- Method for data similarity:

$$\mathcal{O}\left(\left[1+\frac{L}{\mu\sqrt{n}}\right]\log\frac{1}{\varepsilon}\right).$$

- If you have a large amount of data, then this method make $\tilde{\mathcal{O}}(1)$ communications.

Literature

- DANE: Communication Efficient Distributed Optimization using an Approximate Newton-type Method (Shamir et al.)
- DiSCO: DiSCO: Distributed optimization for self-concordant empirical loss (Zhang an Xiao)
- GIANT: GIANT: Globally improved approximate Newton method for distributed optimization (Wang et al.)
- SPAG: Statistically Preconditioned Accelerated Gradient Method for Distributed Optimization (Hendrikx et al.)

Some words about decentralized approach

Decentralized approach

- There is no central server.
- All devices are equivalent and connected in a network.

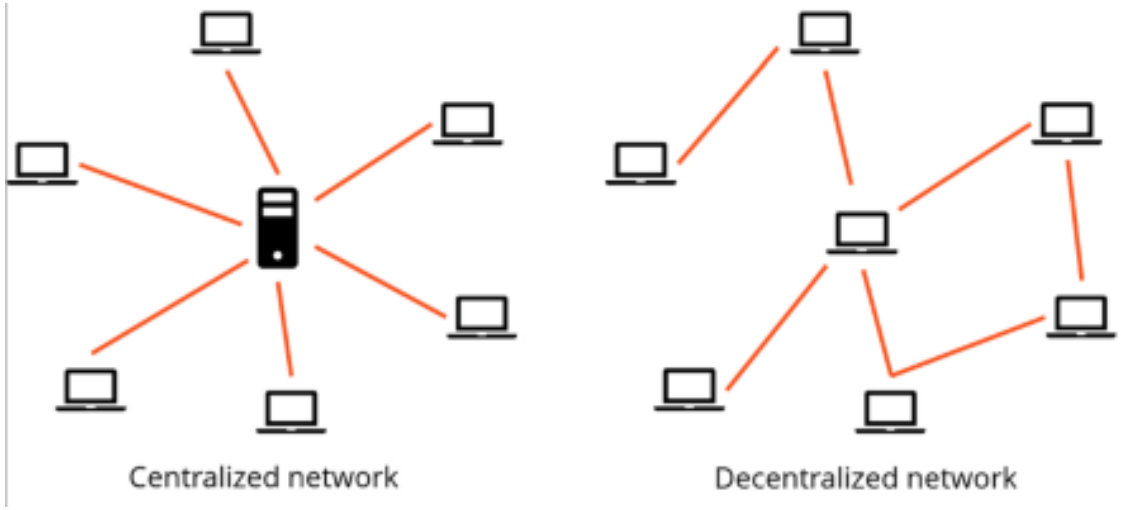

Figure: Centralized and decentralized approaches

When we work in computational clusters:

- We achieve full parallization.
- We can vary network in each iteration: form clusters of devices, select several devices as a server - effectively manage communications.

Decentralized analogues for centralized methods

- Local steps – Local Gossip SGD: A Unified Theory of Decentralized SGD with Changing Topology and Local Updates (Koloskova et al.)
- Quantization – Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication (Koloskova et al.)
- Data Similarity – SONATA: Distributed Optimization Based on Gradient-tracking Revisited: Enhancing Convergence Rate via Surrogation (Sun et al.)

Thank you!