Distributed Machine Learning and Applications

Georgios Kissas

Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania,
Philadelphia, PA, 19104, USA

gkissas@seas.upenn.edu

April 4, 2019

Motivation: The need of distributed machine learning

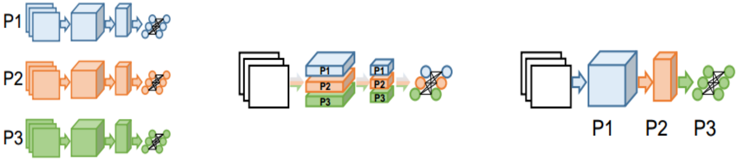
A few reasons for the need of incorporating simulation data into solution algorithms can be:

- Scale matters. Increasing the scale of computations can lead to increased performance.
- In many very complex problems the increased batch size can lead to faster convergence.

How can we increase the scale and the batch size while maintaining performance?

Distributed computing overview: Models

Distributed computing models



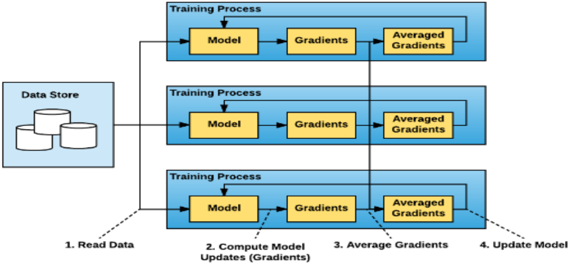
(a) Data Parallelism

(b) Model Parallelism

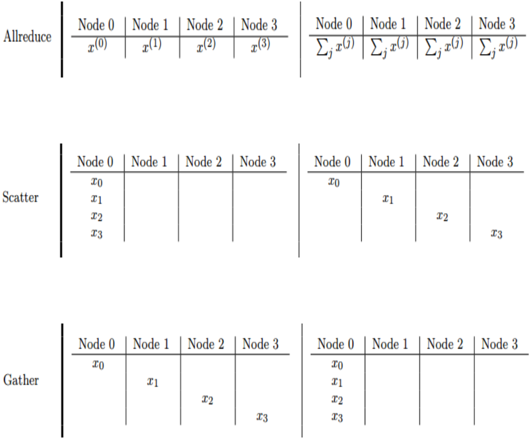
(c) Layer Pipelining

Distributed computing overview: Data parallelization

- Partition the work of the minibatch sample among multiple computational resources
- Forward evaluation and backpropagation operate on a single sample
- The results of the partitions have to be averaged to obtain the gradient w.r.t tha whole minibatch (all reduce operation)
- All parameters must be accessible for all devices, which means the they have to be replicated



Distributed computing overview: Collective communication



Allreduce

Node 0	Node 1	Node 2	Node 3
$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$

Node 0	Node 1	Node 2	Node 3
$\sum_j x^{(j)}$	$\sum_j x^{(j)}$	$\sum_j x^{(j)}$	$\sum_j x^{(j)}$

Scatter

Node 0	Node 1	Node 2	Node 3
x_0			
x_1			
x_2			
x_3			

Node 0	Node 1	Node 2	Node 3
x_0			
	x_1		
		x_2	
			x_3

Gather

Node 0	Node 1	Node 2	Node 3
x_0			
	x_1		
		x_2	
			x_3

Node 0	Node 1	Node 2	Node 3
x_0			
x_1			
x_2			
x_3			

SGD for large mini-batches

The goal is to use large minibatches while maintaining training and generalization accuracy

Linear Scaling Rule: When the minibatch size is multiplied by k, multiply the learning rate by κ.

The update after k iterations for one model:

$$w_{t+k} = w_t - \eta \frac{1}{kn} \sum_{j<k} \sum_{x \in B_j} \nabla L(x, w_{t+j}) \tag{1}$$

The update after 1 iterations for many models:

$$\hat{w}_{t+1} = w_t - \hat{\eta} \frac{1}{kn} \sum_{j<k} \sum_{x \in B_j} \nabla L(x, w_t) \tag{2}$$

The updates for small and large minibatches differ. If we could assume that $L(x, w_t) \approx L(x, w_{t+j})$, by setting $\hat{\eta} = k\eta$ the update would be similar.

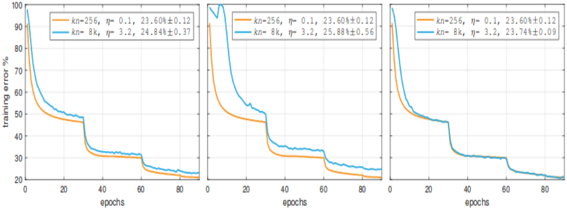
Warm-up phase

The first condition will not hold for two cases:

- In initial training where the training changes rapidly.
- The minibatch size cannot be scaled indefinitely.

To address the first problem they use **warm-up**, a strategy of using less aggressive learning rates at the start of the training.

Gradual warm-up Gradually increases the learning rate from small to large. Start from η and scale to $k\eta$ after a certain amount of epochs.



(a) no warmup (b) constant warmup (c) gradual warmup

Figure 2. **Warmup.** Training error curves for minibatch size 8192 using various warmup strategies compared to minibatch size 256. *Validation* error (mean±std of 5 runs) is shown in the legend, along with minibatch size kn and reference learning rate η.

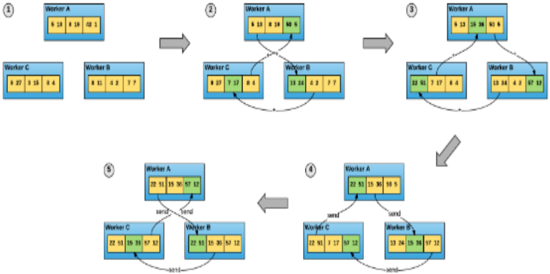
Subtleties and Pitfalls of Distributed SGD

Implementation error **change the definitions** of hyper-parameters, leading to models that train, but the error may be higher than expected. Some straight-forward things to consider:

- **Weight decay**: Scaling the cross-entropy loss is not equivalent with scaling the learning rate.
- **Momentum correction**: Apply momentum correction after changing learning rate if using Momentum SGD.
- **Gradient aggregation**: Normalize the per-worker loss by total minibatch size kn, not per-worker batch-size n.
- **Data shuffling** Use a random shuffling of the training data (per epoch) that is divided amongst all k workers.

Ring-allreduce

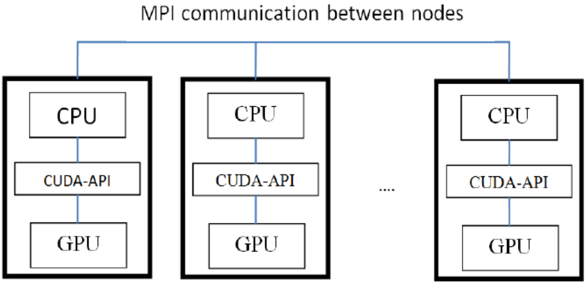
- Each node communicates with the two of it's peers $2 * (N - 1)$ times.In the first $(N - 1)$ iterations, received values are added to the values in the nodes buffer. In the second $(N - 1)$ iterations, received values replace the values held in the nodes buffer.
- Network optimal.
- Allows worker to average gradients and disperse them without using a parameter server.



Hybrid architectures (Horovod)

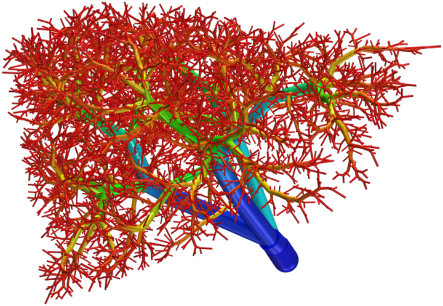
Hybrid architectures adopted in computations:

- Easier to understand an adopt.
- Uses MPI to launch copies to program to other CPUs.
- Sets up the infrastructure for the workers to communicate with each other.

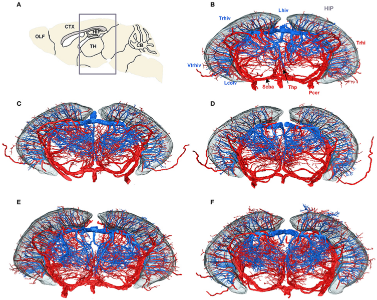


Applications in discovering parameters for arterial networks

If you took all the blood vessels out of an average child and laid them out in one line, the line would stretch over 60,000 miles. An adult's would be closer to 100,000 miles long.



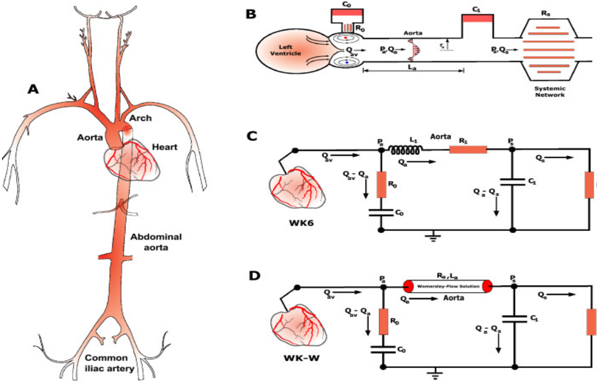
(a) Liver arterial network



(b) Brain arterial network

Applications in discovering parameters for arterial networks

This computationally intractable problem reduces by using a modeling assumption, called the Windkessel model.



Applications in discovering parameters for arterial networks

Physics Informed Deep Learning: Consider a differential equations of the following form:

$$u_t + \mathcal{N}[u] = 0, \quad x \in \Omega, \quad t \in [0, T], \tag{3}$$

where $u(x, t)$ is the solution of the PDE. $\mathcal{N}[\cdot]$ denotes a nonlinear differential operator. $\Omega \in \mathbf{R}^D$ is the spacial domain where the equation holds.

Applications in discovering parameters for arterial networks

In the physics informed neural network, the authors introduce a collocation function of the PDE as:

$$f := u_t + \mathcal{N}[u] \tag{4}$$

Then, they propose the regularization function of the PDEs:

$$\mathrm{MSE} = \mathrm{MSE}_u + \mathrm{MSE}_f \tag{5}$$

where

$$\mathrm{MSE}_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2, \tag{6}$$

and

$$\mathrm{MSE}_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |u(t_u^i, x_f^i)|^2, \tag{7}$$

Where, $\{t_u^i, \mathbf{x}_u^i, u^i\}_{i=1}^{N_u}$ are training data corresponding to the initial conditions and boundary conditions. $\{t_f^i, \mathbf{x}_f^i\}_{i=1}^{N_f}$ are the collocations points within the computational domain.

Applications in discovering parameters for arterial networks

Reduced order model for flow in arterial network:

$$\begin{aligned}\frac{\partial A}{\partial t} + A\frac{\partial u}{\partial x} + u\frac{\partial A}{\partial x} &= 0\\ \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\rho}\frac{\partial p}{\partial x} &= 0\\ p = p_{ext} + \beta(\sqrt{A} - \sqrt{A_0})\end{aligned} \tag{8}$$

Continuity on bifurcations:

$$\begin{aligned}A_1 u_1 &= A_2 u_2 + A_3 u_3\\ p_1 + \frac{1}{2}\rho u_1^2 &= p_2 + \frac{1}{2}\rho u_2^2\\ p_1 + \frac{1}{2}\rho u_1^2 &= p_3 + \frac{1}{2}\rho u_3^2\end{aligned} \tag{9}$$

Boundary conditions:

$$P + R_a C_a \frac{dP}{dt} - (R_a + Z_c)Q - P_{inf} - R_a C_a Z_c \frac{dQ}{dt} = 0 \tag{10}$$

Applications in discovering parameters for arterial networks

By minimizing the loss functions we solve the equation and at the same time discover P, R_a, C_a and Z_c parameters.

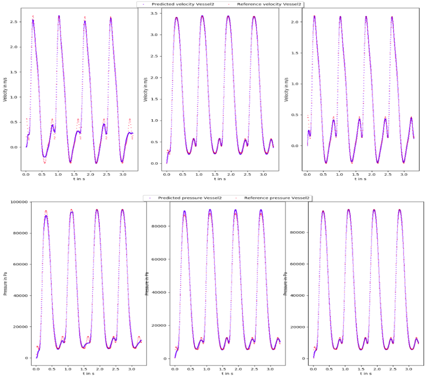


Figure: Velocity and pressure figures for an arterial bifurcation

References

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics Informed Deep Learning (Part I): Data-driven solutions of nonlinear partial differential equations." arXiv preprint arXiv:1711.10561 (2017).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436.

Priya Goyal, Piotr Dollr, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola,Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: TrainingImageNet in 1 hour, 2017, arXiv:1706.02677.

Sergeev, A., Del Balso, M. (2018). Horovod: fast and easy distributed deep learning in TensorFlow. Arxiv.org. Retrieved 13 August 2018, from https://arxiv.org/abs/1802.05799

Raissi, M., Perdikaris, P., Karniadakis, G. E., 2017. Physics informed deep learning (part ii): data-driven discovery of nonlinear partial diferential equations. arXiv preprint arXiv:1711.1056

Sherwin, S., Franke, V., Peiro, J., Parker, K., 2003. One-dimensional modelling of a vascular network in space-time variables. Journal of Engineering Mathematics 47 (3-4), 21725

Thank you for your attention