Research summary in Spring 2020

Yinbin Ma

University of Illinois at Chicago

April 29, 2020

Preface

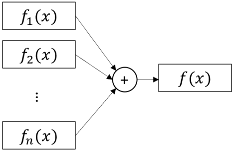
- Topics we covered
  - Decentralized algorithms.
  - Large-scale distributed optimization.
- Tools
  - MPI, Python
- Results
  - Documents and corresponding codes.
- UIC HPC Platform – ACER
  - ACER issued free accounts to students taking Parallel Computing course, each program can be allocated the maximal 7x16 processors.
  - ACER had its own MPI scheduler and compilers.
- Great thanks to Shuo Han!

Table of Contents

Objective Scenario

Many ML problems could be decomposed as following form:

$$\min \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x; A_i) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

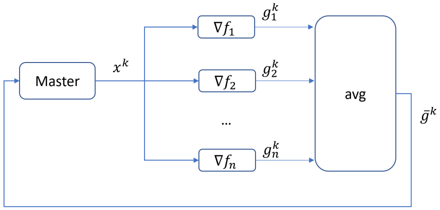


A_i is data used in f_i, f_i is convex, f has a optimal solution x^*.
In this case, we could assign each f_i into a node allocated in a cluster network.

Optimization

We could use Gradient Descent to minimize f:

$$x^{k+1} = x^k - \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(x^k)$$
$$= x^k - \frac{\eta}{n}\sum_{i=1}^{n} g_i^k$$

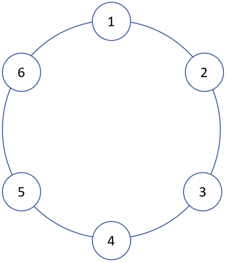


In centralized network, we could ask master node collect gradients from workers, then broadcast updated x.
What if a decentralized network, in other word, how to achieve a average mechanism?

Gossip Matrix

Let's think about d2d networks, each node i has a value to share with its neighbors. Recall the Markov chain, we use a doubly stochastic matrix W representing this "transition", for example a ring.

$$W = \begin{bmatrix} 0.5 & 0.25 & 0 & 0 & 0 & 0.25 \\ 0.25 & 0.5 & 0.25 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.25 & 0 & 0 & 0 & 0.25 & 0.5 \end{bmatrix}$$



$\forall x^0 \in \mathbb{R}^N, x^{k+1} = Wx^t,\ J = \frac{1}{n}\mathbf{1}\mathbf{1}^T$. As $k \to \infty$, $x^k = Jx^0$

Gossip Matrix

Theorem 1

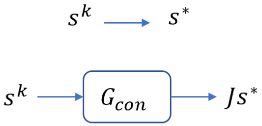
Given a W, if W satisfy following properties,

- $W_{ij} \in [0, 1]$, $W_{ii} > 0$.
- W is symmetry, i.e. $W = W^T$.
- $W \cdot \mathbf{1} = \mathbf{1}$, i.e. $\mathbf{1}$ is an eigenvector of W.
- $\lambda(W) \leq 1$.

Then, W is a gossip matrix, i.e. $\lim_{k\to\infty} x^k = Jx^0$.

Average Consensus Tracking

Suppose that we have a dynamical system G_{con}, G_{con} achieves **average consensus tracking** if we have a sequence $s = \{s^k \in \mathbb{R}^N\}$ converging to s^*, the $\hat{s} = G_{con}(s)$ converge to Js^*.



A example (not unique) of G_{con} is

$$\hat{s}^{k+1} = W\hat{s}^k + (s^{k+1} - s^k), \quad \hat{s}^0 = s^0 \tag{1}$$

Let $\hat{s}^{k+1} = s^{k+1} + \epsilon^{k+1}$, we have

$$\begin{aligned} \epsilon^{k+1} &= \hat{s}^{k+1} - s^{k+1} = W\hat{s}^k - Ws^k + Ws^k - s^k \\ &= W\epsilon^k + (W - I)s^k \end{aligned}$$

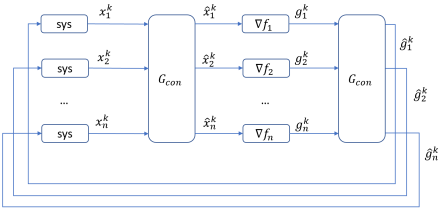
Therefore, (1) could be rewritten as,

$$\hat{s}^{k+1} = s^{k+1} + \epsilon^{k+1}, \quad \epsilon^{k+1} = W\epsilon^k + (W - I)s^k, \quad \epsilon^0 = 0 \tag{2}$$

Average Consensus Tracking

We could use (2) to track x^k and g^k, since both of them will converge to the optimal point, i.e.

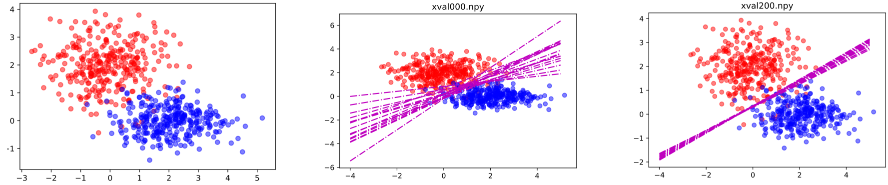
$$x \to x^*, \quad \hat{x} \to Jx^*$$
$$g \to g^*, \quad \hat{g} \to 0$$



Therefore, we succeed to generalize a decentralized gradient descent algorithm.

Average Consensus Tracking

Many optimization algorithms could be rewritten as a decentralized form by using average consensus tracking; here's an example of SVM.



The middle figure shows the optimal decision boundaries without consensus sharing, and the right one is with consensus sharing.

Table of Contents

Stochastic Gradient Descent

Similar to the previous scenario, we assume $\nabla^2 F \succcurlyeq L$ and $\nabla^2 f_i \preccurlyeq \mu$.

$$\min \quad F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x; A_i) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

Another popular method is Stochastic Gradient Descent, it randomly pick one gradient, instead of all gradient to update x.

$$x^{k+1} = x^k - \frac{\eta}{n} \triangledown f_{i_k}(x^k) = x^k - \eta g^k$$

Comparing to GD, SGD decreases the intensity of computation, but it can't guarantee $x^k \to x^*$.

Stochastic Gradient Descent

- $g^k = \nabla f_{i_k}(x^k)$.
- $\mathbb{E}(g^k) = \nabla F(x^k)$.
- $\exists \sigma_g \geq 0,\ c_g \geq 1,\ \mathbb{E}\left[\left\|g^k\right\|_2^2\right] \leq \sigma_g^2 + c_g \left\|\nabla F(x^k)\right\|_2^2$
- $\mathbb{E}\left[F\left(x^k\right) - F\left(x^*\right)\right] \leq \frac{\eta L \sigma_g^2}{2\mu} + (1 - \eta\mu)^t \left(F\left(x^0\right) - F\left(x^*\right)\right)$

Stochastic Gradient Descent

- $g^k = \nabla f_{i_k}(x^k)$.
- $\mathbb{E}(g^k) = \nabla F(x^k)$.
- $\exists \sigma_g \geq 0, c_g \geq 1, \mathbb{E}\left[\left\|g^k\right\|_2^2\right] \leq \sigma_g^2 + c_g \left\|\nabla F(x^k)\right\|_2^2$
- $\mathbb{E}\left[F\left(x^k\right) - F\left(x^*\right)\right] \leq \frac{\eta L \sigma_g^2}{2\mu} + (1-\eta\mu)^t \left(F\left(x^0\right) - F\left(x^*\right)\right)$
- How to reduce the variance of g^k?
    - Find a zero-mean auxiliary variable v and it positively correlated with $\nabla f_{i_k}(x^k)$, i.e. $\left\langle v, \nabla f_{i_k}(x^k)\right\rangle > 0$.
    - Let $\hat{g}^k = \nabla f_{i_k}(x^k) - v, \mathbb{E}(\hat{g}^k) = \nabla F(x^k)$
    - $\mathbb{E}\left[\left\|\hat{g}^k\right\|_2^2\right] = \mathbb{E}\left[\left\|\nabla F(x^k)\right\|_2^2\right] + \mathbb{E}\left[\left\|v\right\|_2^2\right] - 2\mathbb{E}\left[\left\langle v, \nabla f_{i_k}(x^k)\right\rangle\right]$.
- Variance reduction is a way to reduce noise.
    - $\rightarrow$ More accurate estimate $\rightarrow \mathbb{E}\left[F(x^k) - F(x^*)\right] \rightarrow 0$.

Stochastic Variance Reduced Gradient - SVRG

Store the history x^{old} and substitute it for v, i.e.

$$\hat{g}_s^k = \underbrace{\nabla f_{i_k}\left(\boldsymbol{x}_s^k\right) - \nabla f_{i_k}\left(\boldsymbol{x}_s^{\text{old}}\right)}_{\rightarrow \mathbf{0} \text{ if } \boldsymbol{x}_s^k \approx \boldsymbol{x}_s^{\text{old}}} + \underbrace{\nabla F\left(\boldsymbol{x}_s^{\text{old}}\right)}_{\rightarrow \mathbf{0} \text{ if } \boldsymbol{x}_s^{\text{old}} \approx \boldsymbol{x}^*}$$

In s^{th} epoch,

- take snapshots for every $\nabla f_i(x_s^0)$ as $\nabla f_i(x^{\text{old}})$.
- inner loop: for $k = [1, 2, \cdots, m]$
  - $x_s^{k+1} = x_s^k - \eta \hat{g}_s^k$
- Update x_{s+1}^0 for next epoch.

Stochastic Variance Reduced Gradient - SVRG

Store the history x^{old} and substitute it for v, i.e.

$$\hat{g}_s^k = \underbrace{\nabla f_{i_k}\left(\boldsymbol{x}_s^k\right) - \nabla f_{i_k}\left(\boldsymbol{x}_s^{\text{old}}\right)}_{\to \mathbf{0} \text{ if } \boldsymbol{x}_s^k \approx \boldsymbol{x}_s^{\text{old}}} + \underbrace{\nabla F\left(\boldsymbol{x}_s^{\text{old}}\right)}_{\to \mathbf{0} \text{ if } \boldsymbol{x}_s^{\text{old}} \approx \boldsymbol{x}^*}$$

- $\mathbb{E}\left[\left\|\hat{g}^k\right\|_2^2\right] \leq 4L\left[F(x_s^k) - F(x^*) + F(x_s^{\text{old}}) - F(x^*))\right]$
- $\mathbb{E}\left[F(x_s^{\text{old}}) - F(x^*)\right] \leq \rho^s\left[F(x^0) - F(x^*)\right]$
- $\rho = \frac{1}{\mu\eta(1-2L\eta)m} + \frac{2L\eta}{1-2L\eta} < 1$

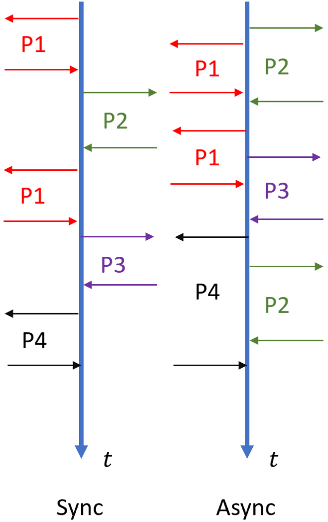
Stochastic Variance Reduced Gradient - SVRG

- Built & tested the SVRG Python program with MPI.
- UIC high performance cluster ACER supports MPI.
- We also succeed built other variance reduction algorithms on ACER, including SAGA.

Table of Contents

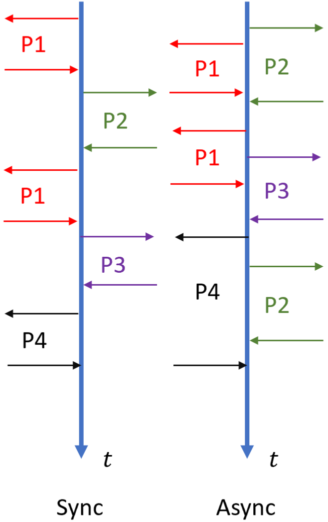
Asynchronous Optimization

- Sync: R/W in interleaved style.
  - Common in analysis.
  - Damage performance a lot.
- Async: R/W in any order.
  - Remove sync lock.
  - Hard to analyze.
  - Achieve linear speedup for parallel scenario using shared memory.



Asynchronous Optimization

- Problems of async:
    - Multiple agents will read the same x.
- Problems of MPI.
    - MPI is built on socket, not shared memory, it already has locks.
    - Only verify if analysis fits practical.
- Still investigating async algorithms, including variance reduction ones.



The End