A Delay-tolerant Proximal-Gradient Algorithm for Distributed Learning

Konstantin Mishchenko  KAUST
Franck Iutzeler  Univ. Grenoble Alpes
Jérôme Malick  CNRS and Univ. Grenoble Alpes
Massih Amini  Univ. Grenoble Alpes

ICML 2018
Global objective

\[
\min_{x \in \mathbb{R}^d} \frac{1}{m} \sum_{j=1}^{m} \ell_j(x) + g(x)
\]

- \(m\) examples
- individual losses \((\ell_j)\)
- empirical risk minimization
- regularizer \(g\)

Local data

\[
\min_{x \in \mathbb{R}^d} \sum_{i=1}^{M} \pi_i f_i(x) + g(x)
\]

- \(M\) data blocks
- stored locally
- local function \((f_i)\)
- \(f_i(x) = \frac{1}{|S_i|} \sum_{j \in S_i} \ell_j(x)\)
- proportion \(\pi_i = |S_i|/m\) at \(i\)

Problem: Large sum minimization
Optimization: Variance-reduced sto. gradient

v.s.

Mid-sized distributed optimization
this presentation
DISTRIBUTED OPTIMIZATION

ASYNCHRONISM

SCARSE COMMUNICATIONS

CONCLUSION
**Problem:** \[ \min_x \sum_{i=1}^{M} \pi_i f_i(x) + g(x) \]

Direct extension of the prox. grad.:

Worker \( i \) update on local variable

\[
 x_i^{k+1/2} = x_i^k - \gamma \nabla f_i(x_i^k)
\]

for all \( i = 1, \ldots, M \)

Master gathering of the local variables

\[
\bar{x}^{k+1} = \sum_{i=1}^{M} \pi_i x_i^{k+1/2}
\]

Master performs a proximity operation

\[
x_1^{k+1} = \ldots = x_M^{k+1} = \text{prox}_{\gamma g} \left( \bar{x}^{k+1} \right)
\]

---

**Implementation Algorithm**

Master:

- Initialize \( \bar{x} = \bar{x}^0 \)
- While not converged do
  - When all workers have finished:
    - Receive \( (x_i) \) from each of them
    - Broadcast \( \bar{x} \) to all agents
    - \( \bar{x} \leftarrow \sum_{i=1}^{M} \pi_i x_i \)
    - Broadcast \( \bar{x} \) to all agents
  - \( k \leftarrow k + 1 \)
- Interrupt all slaves
- Output \( x \)

Worker \( i \):

- Initialize \( x = x_i = \bar{x} \)
- While not interrupted by master do
  - Receive the most recent \( \bar{x} \)
  - \( z \leftarrow \text{prox}_{\gamma g} \left( \bar{x} \right) \)
  - \( x_i \leftarrow z - \gamma \nabla f_i(z) \)
  - Send \( x_i \) to the master

\[
f_i(x) = \frac{1}{|S_i|} \sum_{j \in S_i} \ell_j(x)
\]
**Distributed Proximal Gradient**

**Master:**
- Initialize $\bar{x}$
- **while not converged do**
  - **when all** workers have finished:
    - Receive $(x_i)$ from each of them
    - $\bar{x} \leftarrow \sum_{i=1}^{M} \pi_i x_i$
    - Broadcast $\bar{x}$ to all agents
    - $k \leftarrow k + 1$
- Interrupt all slaves
- **Output** $x$

**Worker $i$:**
- Initialize $x = x_i = \bar{x}$,
- **while not interrupted by master do**
  - Receive the most recent $x$
  - $z \leftarrow \text{prox}_{\gamma g}(\bar{x})$
  - $x_i \leftarrow z - \gamma \nabla f_i(z)$
  - Send $x_i$ to the master

Define time $k$ as the number of master updates

$x^k$ is the value of variable $x$ at time $k$

**Theorem**

Let each $f_i$ be $L$-smooth and $\mu$-strongly convex. Then, for $\gamma \in (0, 2/(\mu + L)]$,

$$
||x^k - x^*||^2 \leq (1 - \alpha)^k ||x^0 - x^*||^2
$$

where $x^*$ is the unique minimizer of the $\min_x \sum_{i=1}^{M} \pi_i f_i(x) + g(x)$ and $\alpha = 2\gamma \mu L / (\mu + L) \in (0, 1]$.

*Proof.* It is exactly proximal gradient descent.
Two Limitations

**Synchronism:** Master waits for *all* workers at each time

<table>
<thead>
<tr>
<th>Agent 1</th>
<th>Agent 2</th>
<th>Agent 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
</tbody>
</table>

image: W. Yin

**Communications:** Sending may be more costly than computing a gradient

Local updates may be:
- *late* (or not, depending on state),
- *fast* (or not, depending on $|S_i|$),
- *costly* (often).

**We provide an efficient Distributed Proximal Gradient algorithm:**

- **Asynchronous** delay-tolerant communications
- **Scarse** comp./comm. tradeoff
DISTRIBUTED OPTIMIZATION

ASYNCHRONISM

SCARSE COMMUNICATIONS

CONCLUSION
Asynchronous Master Slave Framework

Master

\[ x^k = x^{k-1} + \Delta \]

\[ \Delta (i = i(k)) x^k \]

Worker 1

(\( \nabla f_1, \text{prox}_g \))

Worker 1

(\( \nabla f_i, \text{prox}_g \))

Worker M

(\( \nabla f_M, \text{prox}_g \))

\( i = i(k) \) viewpoint

\[ i \quad i \quad i \quad k - D_i^k \quad k = k - d_i^k \]

\( j \neq i(k) \) viewpoint

\[ j \quad j \quad j \quad k - D_j^k \quad k - d_j^k \quad k \]

- **iteration** = receive from a worker + master update + send back
- **time** \( k \) = number of iterations
- **delay** \( d_i^k \) = time since last exchange with \( i \)
  \[ d_i^k = 0 \text{ iff } i \text{ updates at time } k, \quad d_i^k = d_i^{k-1} + 1 \text{ elsewhere} \]
- **second delay** \( D_i^k \) = time since penultimate exchange with \( i \)

**Algorithm** = global communication scheme + local optimization method

what is \( \bar{x} \)

what is \( \bigcirc_i \)
DAve communication scheme
master variable $\bar{x}^k = \text{combination of workers last contributions} (x_i^{k-d_i^k})_i$

one update/time = one worker contribution but all workers are always involved at the master

$$\bar{x}^k = x^{k-1} + \Delta \text{ with } \Delta = \pi_i(x_i^k - x_i^{k-D_i^k}) \text{ for } i = i(k)$$

i.e. $$\bar{x}^k = \sum_{i=1}^{M} \pi_i x_i^{k-d_i^k} = \sum_{i=1}^{M} \pi_i \bigoplus_i (x_i^{k-D_i^k})$$

PG proximal gradient optimization method
one step of proximal gradient on regularizer $g$ and local loss $f_i = \frac{1}{|S_i|} \sum_{j \in S_i} \ell_j$

\[ z \leftarrow \text{prox}(\bar{x}) \]
\[ x_i \leftarrow z - \gamma \nabla f_i(z) \]
\[ \Delta \leftarrow \pi_i (x_i - x_i^{\text{prev}}) \]
\[ x_i^{\text{prev}} \leftarrow x_i \]
DAve-PG

Master:

Initialize $\bar{x}$

while not converged do

when a worker finishes:

Receive adjustment $\Delta$ from it

$\bar{x} \leftarrow \bar{x} + \Delta$

Send $\bar{x}$ to the agent in return

$k \leftarrow k + 1$

Interrupt all slaves

Output $x = \text{prox}_{\gamma g}(\bar{x})$

Worker $i$:

Initialize $x = x_i = \bar{x}$,

while not interrupted by master do

Receive the most recent $\bar{x}$

$z \leftarrow \text{prox}_{\gamma g}(\bar{x})$

$x_i \leftarrow z - \gamma \nabla f_i(z)$

$\Delta \leftarrow \pi_i \left( x_i - x_i^{\text{prev}} \right)$

$x_i^{\text{prev}} \leftarrow x_i$

Send adjustment $\Delta$ to master

$f_i(x) = \frac{1}{|S_i|} \sum_{j \in S_i} \ell_j(x)$

In practice:

- MPI blocking Send and Receive
- No computation/stORAGE at the master
- $x = \text{prox}_{\gamma g}(\bar{x})$ is the converging variable
Comparison with other combinations

<table>
<thead>
<tr>
<th>Combining:</th>
<th>DAve-PG iterates</th>
<th>PIAG gradients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update $x^k$:</td>
<td>$\text{prox}<em>{\gamma g} \left( \sum</em>{i=1}^{M} \pi_i x^k - D_i^k - \gamma \sum_{i=1}^{M} \pi_i \nabla f_i(x^k - D_i^k) \right)$</td>
<td>$\text{prox}<em>{\gamma g} \left( x^{k-1} - \gamma \sum</em>{i=1}^{M} \pi_i \nabla f_i(x^{k-D_i^k}) \right)$</td>
</tr>
</tbody>
</table>

- Combining iterates is more stable than combining gradients
- Example: 2D quadratic functions on 5 worker
  - but one worker 10x slower than the others
  - Stepsize $\gamma$ of PIAG is 10x smaller due to delays
    - the one for DAve-PG stays the same
  - DAve-PG is less chaotic and faster than PIAG

---

Revisiting the clock:

- **epoch sequence** \((k_m)\) = recursively defined by \(k_0 = 0\) and

\[ k_{m+1} = \min\{k : \text{each worker made at least 2 updates on the interval } [k_m, k]\} \]
\[ = \min\{k : k - D_i^k \geq k_m \text{ for all } i = 1, .., M\} \]

- **epoch time** \(m\) = number of epochs

Intuition: \(k_{m+1}\) is the first moment when \(\bar{x}^k\) no longer depends directly on information prior to \(k_m\).

\[ \bar{x}^k = \sum_{i=1}^{M} \pi_i x^{k-D_i^k} - \gamma \sum_{i=1}^{M} \pi_i \nabla f_i(x^{k-D_i^k}) \]

**Theorem**

Let each \(f_i\) be \(L\)-smooth and \(\mu\)-strongly convex. Then, for \(\gamma \in (0, 2/(\mu + L)]\),

\[ \forall k \geq k_m, \quad \|x^k - x^*\|^2 \leq (1 - \alpha)^m \|x^0 - x^*\|^2 \]

where \(x^*\) is the unique minimizer of the \(\min_x \sum_{i=1}^{M} \pi_i f_i(x) + g(x)\) and \(\alpha = 2\gamma L/(\mu + L)\).

Exact same result as the synchronous case but over the **epoch time** \(m\), not \(k\).
Logistic regression w/ elastic net $\frac{1}{m} \sum_{j=1}^{m} \log(1 + \exp(-y_j z_j^T x)) + \lambda_1 ||x||_1 + \frac{\lambda_2}{2} ||x||_2^2$

- 100 machines (1 CPU, 1 GB) in a cluster
- 10% of the data in machine one, even on the rest
Distributed Optimization

Asynchronism

Scarse Communications

Conclusion
To exchange less, a solution is to compute more.

**DAve-RPG**

**Master:**
- Initialize $\bar{x}$
- **while** not converged **do**
  - **when** a worker finishes:
    - Receive adjustment $\Delta$ from it
    - $\bar{x} \leftarrow \bar{x} + \Delta$
    - Send $\bar{x}$ to the agent in return
  - $k \leftarrow k + 1$
- Interrupt all slaves
- **Output** $x = \text{prox}_\gamma g(\bar{x})$

**Worker $i$:**
- Initialize $x = x_i = \bar{x}$,
- **while** not interrupted by master **do**
  - Receive the most recent $\bar{x}$
  - Select a number of repetitions $p$
  - Initialize $\Delta = 0$
  - **for** $q = 1$ to $p$ **do**
    - $z \leftarrow \text{prox}_\gamma g(\bar{x} + \Delta)$
    - $x_i \leftarrow z - \gamma \nabla f_i(z)$
    - $\Delta \leftarrow \Delta + \pi_i(x_i - x_i^{\text{prev}})$
    - $x_i \leftarrow x_i^{\text{prev}}$
  - Send the adjustment $\Delta$ to the master

$f_i(x) = \frac{1}{|S_i|} \sum_{j \in S_i} \ell_j(x)$

**Difference with before:**
- **at each** local step, the worker performs $p$ proximal gradient steps
- controlled rate improvement by

$$\max_{\text{repetitions } p \text{ in the epoch}} 1 - \frac{\gamma \mu}{\sum_{q=1}^{p-1} (1 - \gamma \mu)^{q-1} \min_i \pi_i^q}$$

**but** the epochs become longer
- Logistic regression w/ elastic net: \( \frac{1}{m} \sum_{j=1}^{m} \log(1 + \exp(-y_j z_j^T x)) + \lambda_1 \|x\|_1 + \frac{\lambda_2}{2} \|x\|_2^2 \)

- 100 machines (1 CPU, 1 GB) in a cluster

- 10% of the data in machine one, even on the rest

- there is a compromise to find ...

- ... but \( p \) can be changed without restrictions
DISTRIBUTED OPTIMIZATION

ASYNCHRONISM

SCARSE COMMUNICATIONS

■ CONCLUSION
Distributed Delay-Tolerant Proximal Gradient Algorithm:

- Simple to implement
- Adaptable to performance/computation compromise
- General, adaptable epoch analysis

Poster # 155

Future works:

- Sparse communications
- Using identification to control the communications

Thank you!  – Franck IUTZELER  http://www.iutzeler.org