AIDE: Fast and Communication Efficient Distributed Optimization

Sashank J. Reddi*, Jakub Konečný^, Peter Richtárik^, Barnabás Póczos*, Alex Smola*

*Machine Learning Department, Carnegie Mellon University
^School of Mathematics, University of Edinburgh

Distributed Optimization Problem
Minimize an average of functions, each of which is stored on different machine. Formally,
\[ \min_{w \in \mathbb{R}^d} f(w) := \frac{1}{K} \sum_{k=1}^{K} F_k(w) \]
where \( K \) is the number of machines, and
\[ F_k(w) = \frac{1}{N} \sum_{i \in P_k} f_i(w) \]
Here, \( f_i \) usually represent a loss function incurred on the i-th data point. Set \( P_k \) denotes indices of data points stored on computer \( k \).

Baseline algorithm: DANE [3]
At iteration \( t \), with iterate \( w^{t-1} \), each machine solves
\[ \tilde{w}^{t}_{k} = \arg \min_{w \in \mathbb{R}^d} g^{t}_{k,m}(w), \]
where
\[ g^{t}_{k,m}(w) := F_k(w) - \frac{\alpha}{2} \left\| w - w^{t-1} \right\|^2. \]
This is followed by aggregation to form new iterate
\[ w^{t} = \frac{1}{K} \sum_{k=1}^{K} w^{t}_{k}. \]

Properties
- Fast convergence when \( F_k(w) \) are similar enough (i.i.d. data distribution)
- Not robust to arbitrary data distributions
- Exact minimum computationally infeasible in many applications

Our contributions
- Inexact version of DANE – more robust
- Accelerated version, AIDE, that (nearly) matches communication complexity lower bounds
- First efficient method that can be implemented using only first-order oracle

Two Distributed Optimization Algorithms

Inexact DANE
Core idea: solve the DANE subproblem approximately.

Algorithm 1: InexactDANE(\( f, w^{0}, \gamma, \alpha, \tau \))
\[ \text{Inputs: } f(w) = \frac{1}{K} \sum_{k=1}^{K} F_k(w), \text{ initial point } w^{0} \in \mathbb{R}^d, \text{ inexactness parameter } 0 \leq \gamma < 1 \]
\[ \text{for } t = 1 \text{ to } \infty \text{ do} \]
\[ \text{for } k = 1 \text{ to } K \text{ do in parallel} \]
\[ \quad \text{Find an approximate solution } w^{t}_{k} := \arg \min_{w \in \mathbb{R}^d} g^{t}_{k,m}(w), \text{ such that } \left\| w^{t}_{k} - w^{t-1}_{k} \right\| / \left\| w^{t-1} \right\| \leq \gamma \]
\[ \text{end} \]
\[ w^{t} = \frac{1}{K} \sum_{k=1}^{K} w^{t}_{k} \]
\[ \text{return } w^{t} \]

AIDE: Accelerated Inexact DANE
Core idea: apply Universal Catalyst [1] to InexactDANE

Algorithm 1: AIDE(\( f, w^{0}, \lambda, \gamma, \alpha, \tau, \epsilon \))
\[ \text{Inputs: } f(w) = \frac{1}{K} \sum_{k=1}^{K} F_k(w), \text{ initial point } w^{0} \in \mathbb{R}^d, \text{ inexactDANE iterations } s, \text{ inexactness parameter } 0 \leq \gamma < 1, \tau \geq 0, \text{ Let } \rho = \lambda / (\lambda + \tau) \]
\[ \text{while } f(w^{t-1}) - f(w^{t}) \leq \epsilon \text{ do} \]
\[ \quad \text{Define } f'(w) := \frac{1}{K} \sum_{k=1}^{K} F_k(w) + \frac{\alpha}{2} \left\| w - w^{t-1} \right\|^2 \]
\[ w^{t} = \text{InexactDANE}(f', \tilde{w}^{t-1}, s, \gamma, \rho) \]
\[ \text{Find } \tilde{z} \in (0, 1) \text{ such that } \tilde{z}^2 = (1 - \epsilon)/\epsilon \]
\[ \text{Compute } y^{t} = w^{t} + \tilde{z}(w^{t} - w^{t-1}) \text{ where } \delta_{t} = \frac{\lambda^{t} / (\lambda + \tau)}{\tilde{z}^2 + \rho} \]
\[ \text{return } y^{t} \]

Communication complexity guarantees

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( \delta )-related ( F_k )</th>
<th>strongly convex ( F_k )</th>
<th>convex ( F_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>InexactDANE</td>
<td>( O\left( \frac{\sqrt{\delta}}{\log | \epsilon |} \right) )</td>
<td>( O\left( \frac{\delta}{\log | \epsilon |} \right) )</td>
<td>( O\left( \frac{\delta}{\log | \epsilon |} \right) )</td>
</tr>
<tr>
<td>AIDE</td>
<td>( O\left( \frac{\sqrt{\delta}}{\log | \epsilon |} \right) )</td>
<td>( O\left( \frac{\delta}{\log | \epsilon |} \right) )</td>
<td>( O\left( \frac{\delta}{\log | \epsilon |} \right) )</td>
</tr>
</tbody>
</table>

\( \lambda \) – strong convexity; \( I \) – smoothness parameter; \( \epsilon \) – target accuracy

- Only modest accuracy necessary for the above rates (\( \gamma \approx 1/8 \))
- Inexactness changes only constants and adds practical robustness

Experiments
In the following, compare three algorithms:
- COCOA ([2]) with SDCA locally
- InexactDANE (DANE) with SVRG locally
- AIDE with SVRG locally

Node scaling
- Rcv1 dataset, smoothed hinge loss
- Regularization strength \( \{ 1/10, 1/100 \} \)
- Data randomly distributed across 8 nodes.

Arbitrary data partitioning
- Rcv1, covtype, realism, url datasets, logistic loss
- Fixed number of local steps of SVRG
- Partitioned to 2 computers:
  - Randomly (random)
  - Based on output label (output)

References