Randomized Primal-Dual Algorithms for Asynchronous Distributed Optimization

Lin Xiao
Microsoft Research

Joint work with
Adams Wei Yu (CMU), Qihang Lin (University of Iowa)
Weizhu Chen (Microsoft)

Workshop on Large-Scale and Distributed Optimization
Lund Center for Control of Complex Engineering Systems
June 14-16, 2017
Motivation

big data optimization problems

- dataset cannot fit into memory or storage of single computer
- require distributed algorithms with inter-machine communication

origins

- machine learning, data mining, ...
- industry: search, online advertising, social media analysis, ...

goals

- asynchronous distributed algorithms deployable in the cloud
- nontrivial communication and/or computation complexity
Outline

• distributed empirical risk minimization

• randomized primal-dual algorithms with parameter servers

• variance reduction techniques

• DSCOVR algorithms  
  (Doubly Stochastic Coordinate Optimization with Variance Reduction)

• preliminary experiments
Empirical risk minimization (ERM)

- popular formulation in supervised (linear) learning

\[
\begin{align*}
\text{minimize} \quad & P(w) \overset{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} \phi(x_i^T w, y_i) + \lambda g(w) \\
\text{where} \quad & x_i \in \mathbb{R}^d, \quad y_i \in \mathbb{R}
\end{align*}
\]

- i.i.d. samples: \((x_1, y_1), \ldots, (x_N, y_N)\) where \(x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\)
- loss function: \(\phi(\cdot, y)\) convex for every \(y\)
- \(g(w)\) strongly convex, e.g., \(g(w) = (\lambda/2)\|w\|_2^2\)
- regularization parameter \(\lambda \sim 1/\sqrt{N}\) or smaller

- **linear regression**: \(\phi(x^T w, y) = (y - w^T x)^2\)

- **binary classification**: \(y \in \{\pm 1\}\)
  - logistic regression: \(\phi(x^T w, y) = \log(1 + \exp(-y(w^T x)))\)
  - hinge loss (SVM): \(\phi(x^T w, y) = \max\{0, 1 - y(w^T x)\}\)
Distributed ERM

when dataset cannot fit into memory of single machine

• data partitioned on \( m \) machines

\[
X = \begin{bmatrix}
    x_1^T \\
    x_2^T \\
    \vdots \\
    x_N^T
\end{bmatrix} \in \mathbb{R}^{N \times d}
\]

• rewrite objective function

\[
\minimize_{w \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{m} \Phi_i(X_i; w) + g(w)
\]

where \( \Phi_i(X_i; w) = \sum_{j \in I_i} \phi_j(x_j^T w, y_j) \) and \( \sum_{i=1}^{m} |I_i| = N \)
Distributed optimization

- **distributed algorithms:** alternate between
  - a local computation procedure at each machine
  - a communication round with simple map-reduce operations
    (e.g., broadcasting a vector in \( \mathbb{R}^d \) to \( m \) machines, or computing sum or average of \( m \) vectors in \( \mathbb{R}^d \))

- **bottleneck:** high cost of inter-machine communication
  - speed/delay, synchronization
  - energy consumption

- **communication-efficiency**
  - number of communication rounds to find \( P(\hat{w}) - P(w^*) \leq \epsilon \)
  - often can be measured by iteration complexity
**Iteration complexity**

- **assumption:** \( f : \mathbb{R}^d \to \mathbb{R} \) twice continuously differentiable,
  \[
  \lambda I \preceq f''(w) \preceq LI, \quad \forall w \in \mathbb{R}^d
  \]
  in other words, \( f \) is \( \lambda \)-strongly convex and \( L \)-smooth

- **condition number**
  \[
  \kappa = \frac{L}{\lambda}
  \]
  we focus on ill-conditioned problems: \( \kappa \gg 1 \)

- **iteration complexities** of first-order methods
  - gradient descent method: \( \mathcal{O}(\kappa \log(1/\epsilon)) \)
  - accelerated gradient method: \( \mathcal{O}(\sqrt{\kappa} \log(1/\epsilon)) \)
  - stochastic gradient method: \( \mathcal{O}(\kappa/\epsilon) \) (population loss)
Distributed gradient methods

distributed implementation of gradient descent

\[
\minimize_{w \in \mathbb{R}^d} \quad P(w) = \frac{1}{N} \sum_{i=1}^{m} \Phi_i(X_i; w)
\]

- each iteration involves one round of communication
- number of communication rounds: \( O(\kappa \log(1/\epsilon)) \)
- can use accelerated gradient method: \( O(\sqrt{\kappa} \log(1/\epsilon)) \)
ADMM

- reformulation: \[
\text{minimize} \quad \frac{1}{N} \sum_{i=1}^{m} f_i(u_i) \\
\text{subject to} \quad u_i = w, \quad i = 1, \ldots, m
\]

- augmented Lagrangian

\[
L_\rho(u, v, w) = \sum_{i=1}^{m} \left( f_i(u_i) + \langle v_i, u_i - w \rangle + \frac{\rho}{2} \| u_i - w \|_2^2 \right)
\]

\[
\begin{align*}
w^{(t+1)} &= \arg \min_w L_\rho(u^{(t+1)}, v^{(t)}, w) \\
\text{communicate } O(d) \text{ bits}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{master} \\
\uparrow \quad \downarrow \\
\uparrow \quad \downarrow \\
1 \quad \cdots \quad i \quad \cdots \quad m \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
u_i^{(t+1)} &= \arg \min_{u_i} L_\rho(u_i, v^{(t)}, w^{(t)}) \\
v_i^{(t+1)} &= v_i^{(t)} + \rho(u_i^{(t)} - w^{(t)})
\end{align*}
\]

- no. of communication rounds: \(O(\kappa \log(1/\epsilon))\) or \(O(\sqrt{\kappa} \log(1/\epsilon))\)
The dual ERM problem

primal problem

\[
\min_{w \in \mathbb{R}^d} \quad P(w) \overset{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{m} \Phi_i(X_i;w) + g(w)
\]

dual problem

\[
\max_{\alpha \in \mathbb{R}^N} \quad D(\alpha) \overset{\text{def}}{=} -\frac{1}{N} \sum_{i=1}^{m} \Phi_i^*(\alpha_i) - g^* \left( -\frac{1}{N} \sum_{i=1}^{m} (X_i:)^T \alpha_i \right)
\]

where \( g^* \) and \( \phi_i^* \) are convex conjugate functions

- \( g^*(v) = \sup_{u \in \mathbb{R}^d} \{ v^T u - g(u) \} \)
- \( \Phi_i^*(\alpha_i) = \sup_{z \in \mathbb{R}^{n_i}} \{ \alpha_i^T z - \Phi_i(z) \} \), for \( i = 1, \ldots, m \)

recover primal variable from dual: \( w = \nabla g^* \left( -\frac{1}{N} \sum_{i=1}^{m} (X_i:)^T \alpha_i \right) \)
The CoCoA(+) algorithm

(Jaggi et al. 2014, Ma et al. 2015)

\[
\begin{align*}
\text{maximize } & \quad D(\alpha) \overset{\text{def}}{=} -\frac{1}{N} \sum_{i=1}^{m} \Phi_i^*(\alpha_i) - g^* \left( -\frac{1}{N} \sum_{i=1}^{m} (X_i:)^T \alpha_i \right) \\

\text{master} & \\
1 & \cdots & i & \cdots & m \\
\end{align*}
\]

\[
\nu^{(t+1)} = \nu^{(t)} + \sum_{i=1}^{m} \Delta \nu_i^{(t)}
\]

communicate \( O(d) \) bits

\[
\begin{align*}
\alpha_i^{(t+1)} &= \arg \max_{\alpha_i} G_i(\nu^{(t)}, \alpha_i) \\
\Delta \nu_i^{(t)} &= \frac{1}{N} (X_i:)^T (\alpha_i^{(t+1)} - \alpha_i^{(t)})
\end{align*}
\]

- each iteration involves one round of communication
- number of communication rounds: \( O(\kappa \log(1/\epsilon)) \)
- can be accelerated by PPA (Catalyst, Lin et al.): \( O(\sqrt{\kappa} \log(1/\epsilon)) \)
Primal and dual variables

\[ w = \nabla g^* \left( -\frac{1}{N} \sum_{i=1}^{m} (X_i:)^T \alpha_i \right) \]
Can we do better?

- asynchronous distributed algorithms?
- better communication complexity?
- better computation complexity?
Outline

- distributed empirical risk minimization
- randomized primal-dual algorithms with parameter servers
- variance reduction techniques
- DSCOVR algorithms
  (Doubly Stochastic Coordinate Optimization with Variance Reduction)
- preliminary experiments
Asynchronism: Hogwild! style

**idea:** exploit sparsity to avoid simultaneous updates (Niu et al. 2011)

![Diagram of machine communication](image.png)

**problems:**
- too frequent communication (bottleneck for distributed system)
- slow convergence (sublinear rate rate using stochastic gradients)
Tame the hog: forced separation

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_i$</th>
<th>$w_K$</th>
</tr>
</thead>
</table>

machine 1

machine 2

machine $m$

• partition $w$ into $K$ blocks $w_1, \ldots, w_K$
• each machine updates a different block using relevant columns
• set $K > m$ so that all machines can work all the time
• event-driven asynchronism:
  – whenever free, each machine request new block to update
  – update orders can be intentionally randomized
Double separation via saddle-point formulation

\[
\min_{w \in \mathbb{R}^d} \max_{\alpha \in \mathbb{R}^N} \left\{ \frac{1}{N} \sum_{i=1}^{m} \sum_{k=1}^{K} \alpha_i^T X_{ik} w_k - \frac{1}{N} \sum_{i=1}^{m} \Phi_i^*(\alpha_i) + \sum_{k=1}^{K} g(w_k) \right\}
\]
A randomized primal-dual algorithm

**Algorithm 1**: Doubly stochastic primal-dual coordinate update

**input**: initial points $w^{(0)}$ and $\alpha^{(0)}$

**for** $t = 0, 1, 2, \ldots, T - 1$

1. pick $j \in \{1, \ldots, m\}$ and $l \in \{1, \ldots, K\}$ with probabilities $p_j$ and $q_l$

2. compute stochastic gradients

$$u_j^{(t+1)} = \frac{1}{q_l} X_{jl} w_l^{(t)}; \quad v_l^{(t+1)} = \frac{1}{p_j} \frac{1}{N} (X_{jl})^T \alpha_j^{(t)}$$

3. update primal and dual block coordinates:

$$\alpha_i^{(t+1)} = \begin{cases} \text{prox}_{\sigma_j \psi_j^*} (\alpha_j^{(t)} + \sigma_j u_j^{(t+1)}) & \text{if } i = j, \\ \alpha_i^{(t)} & \text{if } i \neq j, \end{cases}$$

$$w_k^{(t+1)} = \begin{cases} \text{prox}_{\tau_l \phi_l} (w_l^{(t)} - \tau_l v_l^{(t+1)}) & \text{if } k = l, \\ w_k^{(t)} & \text{if } k \neq l. \end{cases}$$

**end for**
How good is this algorithm?

- on the update order
  - sequence $(i(t), k(t))$ not really i.i.d.
  - in practice better than i.i.d.?

- machine 1
- machine 2
- machine $m$

- bad news: sublinear convergence, with complexity $O(1/\epsilon)$
Outline

• distributed empirical risk minimization

• randomized primal-dual algorithms with parameter servers

• variance reduction techniques

• DSCOVR algorithms
  (Doubly Stochastic Coordinate Optimization with Variance Reduction)

• preliminary experiments
Minimizing finite average of convex functions

\[
\text{minimize} \quad F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) + g(w)
\]

• **batch proximal gradient method**
  \[
  w^{(t+1)} = \text{prox}_{\eta_t g} \left( w^{(t)} - \eta_t \nabla F(w^{(t)}) \right)
  \]
  – each step very expensive, relatively fast convergence
  – can use accelerated proximal gradient methods

• **stochastic proximal gradient method**
  \[
  w^{(t+1)} = \text{prox}_{\eta_t g} \left( w^{(t)} - \eta_t \nabla f_{i_t}(w^{(t)}) \right) \quad (i_t \text{ chosen randomly})
  \]
  – each iteration very cheap, but very slow convergence
  – accelerated stochastic algorithms do not really help

• recent advances in **randomized algorithms**: exploit finite average (sum) structure to get best of both worlds
Stochastic variance reduced gradient (SVRG)

- SVRG (Johnson & Zhang 2013)
  - update form
    \[ w^{(t+1)} = w^{(t)} - \eta (\nabla f_{i_t}(w^{(t)}) - \nabla f_{i_t}(\tilde{w}) + \nabla F(\tilde{w})) \]
  - update \( \tilde{w} \) periodically (every few passes)
- still a stochastic gradient method
  \[ \mathbf{E}_{i_t}[\nabla f_{i_t}(w^{(t)}) - \nabla f_{i_t}(\tilde{w}) + \nabla F(\tilde{w})] = \nabla F(w^{(t)}) \]
  - expected update direction is the same as \( \mathbf{E}[\nabla f_{i_t}(w^{(t)})] \)
  - variance can be diminishing if \( \tilde{w} \) updated periodically
- complexity: \( O \left( (n + \kappa) \log \frac{1}{\epsilon} \right) \), cf. SGD \( O(\kappa/\epsilon) \)
- Prox-SVRG (X. and Zhang 2014): same complexity
Intuition of variance reduction

\[ \nabla f_{i_t}(\tilde{w}) - \nabla F(\tilde{w}) - \nabla f_{i_t}(\tilde{w}) \]

\[ \nabla f_{i_t}(w^{(t)}) - \nabla F(\tilde{w}) - \nabla f_{i_t}(\tilde{w}) \]
SAGA (Defazio, Bach & Lacoste-Julien 2014)

- the algorithm

\[ w^{(t+1)} = w^{(t)} - \eta_t \left[ \nabla f_{i_t}(w^{(t)}) - \nabla f_{i_t}(z_{i_t}^{(t)}) + \frac{1}{n} \sum_{j=1}^{n} \nabla f_j(z_j^{(t)}) \right] \]

\( z_{j}^{(t)} \): last point at which component gradient \( \nabla f_{j} \) was calculated

- naturally extends to proximal version

- complexity: \( O \left( (n + \kappa) \log \frac{1}{\varepsilon} \right) \), cf. SGD \( O(\kappa / \varepsilon) \)
## Condition number and batch complexity

- **Condition number**: $\kappa = \frac{R^2}{\lambda \gamma}$ (considering $\kappa \gg 1$)
- **Batch complexity**: number of equivalent passes over dataset complexities to reach $\mathbb{E}[P(w(t)) - P^*] \leq \epsilon$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Iteration complexity</th>
<th>Batch complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic gradient</td>
<td>$(1 + \kappa)/\epsilon$</td>
<td>$(1 + \kappa)/(n\epsilon)$</td>
</tr>
<tr>
<td>Full gradient (FG)</td>
<td>$(1 + \kappa')\log(1/\epsilon)$</td>
<td>$(1 + \kappa')\log(1/\epsilon)$</td>
</tr>
<tr>
<td>Accelerated FG (Nesterov)</td>
<td>$(1 + \sqrt{\kappa'})\log(1/\epsilon)$</td>
<td>$(1 + \sqrt{\kappa'})\log(1/\epsilon)$</td>
</tr>
<tr>
<td>SDCA, SAG(A), SVRG, ...</td>
<td>$(n + \kappa)\log(1/\epsilon)$</td>
<td>$(1 + \kappa/n)\log(1/\epsilon)$</td>
</tr>
<tr>
<td>A-SDCA, APCG, SPDC, ...</td>
<td>$(n + \sqrt{\kappa n})\log(1/\epsilon)$</td>
<td>$(1 + \sqrt{\kappa/n})\log(1/\epsilon)$</td>
</tr>
</tbody>
</table>

**Algorithm References**

- SDCA: Shalev-Shwartz & Zhang (2013)
- Finito: Defazio, Caetano & Domke (2014)
- Quartz: Qu, Richtárik, & Zhang (2015)
- Catalyst: Lin, Mairal, & Harchaoui (2015)
- RPDG: Lan (2015)

- SAGA: Defazio, Bach & Lacoste-Julien (2014)
- A-SDCA: Shalev-Shwartz & Zhang (2014)
- APCG: Lin, Lu & X. (2014)

**Lower Bound References**

Outline

- distributed empirical risk minimization
- randomized primal-dual algorithms with parameter servers
- variance reduction techniques
- **DSCOVR algorithms**
  (Doubly Stochastic Coordinate Optimization with Variance Reduction)
- preliminary experiments
Double separation via saddle-point formulation

\[ \min_{w \in \mathbb{R}^d} \max_{\alpha \in \mathbb{R}^N} \left\{ \frac{1}{N} \sum_{i=1}^{m} \sum_{k=1}^{K} \alpha_i^T X_{ik} w_k - \frac{1}{N} \sum_{i=1}^{m} \Phi_i^*(\alpha_i) + \sum_{k=1}^{K} g(w_k) \right\} \]
Algorithm 2: DSCOVR-SVRG

for $s = 0, 1, 2, \ldots, S - 1$

- $\bar{u}^{(s)} = X \bar{w}^{(s)}$ and $\bar{v}^{(s)} = \frac{1}{N} X^T \bar{\alpha}^{(s)}$
- $w^{(0)} = \bar{w}^{(s)}$ and $\alpha^{(0)} = \bar{\alpha}^{(s)}$

- for $t = 0, 1, 2, \ldots, T - 1$
  1. pick $j \in \{1, \ldots, m\}$ and $l \in \{1, \ldots, K\}$ with probabilities $p_j$ and $q_l$
  2. compute variance-reduced stochastic gradients:

$$u_j^{(t+1)} = \bar{u}_j^{(s)} + \frac{1}{q_l} X_{jl} (w_l^{(t)} - \bar{w}_l^{(s)}), \quad v_l^{(t+1)} = \bar{v}_l^{(s)} + \frac{1}{p_j} \frac{1}{N} (X_{jl})^T (\alpha_j^{(t)} - \bar{\alpha}_j^{(s)})$$

  3. update primal and dual block coordinates:

$$\alpha_i^{(t+1)} = \begin{cases} \text{prox}_{\sigma_j \psi_j^*} (\alpha_j^{(t)} + \sigma_j u_j^{(t+1)}) & \text{if } i = j, \\ \alpha_i^{(t)} & \text{if } i \neq j, \end{cases}$$

$$w_k^{(t+1)} = \begin{cases} \text{prox}_{\tau_l g_l} (w_l^{(t)} - \tau_l v_l^{(t+1)}) & \text{if } k = l, \\ w_k^{(t)} & \text{if } k \neq l. \end{cases}$$

end for

- $\bar{w}^{(s+1)} = w^{(T)}$ and $\bar{\alpha}^{(s+1)} = \alpha^{(T)}.$

end for
Convergence analysis of DSCOVR-SVRG

• assumptions:
  – each $\phi_i$ is $1/\gamma$-smooth $\implies$ $\phi_i^*$ is $\gamma$-strongly convex

\[
|\phi_i'(\alpha) - \phi_i'(\beta)| \leq \frac{1}{\gamma} |\alpha - \beta|, \quad \forall \alpha, \beta \in \mathbb{R}
\]

  – $g$ is $\lambda$-strongly convex $\implies$ $g^*$ is $1/\lambda$-smooth

\[
g(w) \geq g(u) + g'(u)^T(w - u) + \frac{\lambda}{2} \|w - u\|_2^2, \quad \forall w, u \in \mathbb{R}^d
\]

• strong duality
  – there exist unique $(w^*, \alpha^*)$ satisfying $P(w^*) = D(\alpha^*)$
  – $w^* = \nabla g^*\left( -\frac{1}{N} \sum_{i=1}^{m} (X_i:)^T \alpha_i^* \right)$
**Theorem:** Let $\Lambda$ and $\Gamma$ be two constants that satisfy

$$\Lambda \geq \|X_{ik}\|^2, \quad \text{for all } i = 1, \ldots, m, \text{ and } j = 1, \ldots, K,$$

$$\Gamma \geq \max_{i,k} \left\{ \frac{1}{p_i} \left( 1 + \frac{9m\Lambda}{2q_k N\lambda\gamma} \right), \frac{1}{q_k} \left( 1 + \frac{9K\Lambda}{2p_i N\lambda\gamma} \right) \right\}.$$

If we choose the step sizes as

$$\sigma_i = \frac{1}{2\gamma (p_i \Gamma - 1)}, \quad i = 1, \ldots, m,$$

$$\tau_k = \frac{1}{2\lambda (q_k \Gamma - 1)}, \quad k = 1, \ldots, K,$$

and the number of iterations during each stage $T \geq \log(3)\Gamma$, then

$$\mathbb{E} \left[ \left\| \tilde{w}^{(s)} - w^* \right\|^2_{\lambda, \frac{\gamma}{N}} \right] \leq \left( \frac{2}{3} \right)^s \left\| \tilde{w}^{(0)} - w^* \right\|^2_{\lambda, \frac{\gamma}{N}}$$
Complexity analysis (assuming $K > m$)

- if $p_i = \frac{1}{m}$ and $q_k = \frac{1}{K}$, then can take $\Gamma = K \left(1 + \frac{9mK\Lambda}{2N\lambda\gamma}\right)$
- if $p_i = \frac{\|X_i\|_F^2}{\|X\|_F^2}$ and $q_k = \frac{\|X_k\|_F^2}{\|X\|_F^2}$, then $\Gamma = K \left(1 + \frac{9\|X\|_F^2}{2N\lambda\gamma}\right)$
- if $\max_i \|x_i\| \leq R$, then can use $\Gamma = K \left(1 + \frac{9R^2}{2\lambda\gamma}\right) = K \left(1 + \frac{9}{2}\kappa\right)$

**complexities**

- iteration complexity (number of $X_{ik}$ blocks processed):
  \[ O \left(K\left(1 + m + \kappa\right)\log\frac{1}{\epsilon}\right) \]
- communication complexity (number of $d$-vectors transmitted):
  \[ O \left((1 + m + \kappa)\log\frac{1}{\epsilon}\right) \]
- computation complexity (number of passes over whole dataset):
  \[ O \left((1 + \frac{\kappa}{m})\log\frac{1}{\epsilon}\right) \]
Convergence of duality gap

**Theorem:** Let $\Lambda$ and $\Gamma$ be two constants that satisfy

\[
\Lambda \geq \|X_{ik}\|^2, \quad \text{for all} \quad i = 1, \ldots, m, \text{ and } j = 1, \ldots, K,
\]

\[
\Gamma \geq \max_{i,k} \left\{ \frac{1}{p_i} \left(1 + \frac{18m\Lambda}{q_k N\lambda\gamma}\right), \frac{1}{q_k} \left(1 + \frac{18K\Lambda}{p_i N\lambda\gamma}\right) \right\}.
\]

If we choose the step sizes as

\[
\sigma_i = \frac{1}{\gamma(p_i\Gamma - 1)}, \quad i = 1, \ldots, m,
\]

\[
\tau_k = \frac{1}{\lambda(q_k\Gamma - 1)}, \quad k = 1, \ldots, K,
\]

and the number of iterations during each stage $T \geq \log(3)\Gamma$, then

\[
E \left[ P(\tilde{w}^{(s)}) - D(\tilde{\alpha}^{(s)}) \right] \leq \left(\frac{2}{3}\right)^s 3\Gamma \left( P(\tilde{w}^{(0)}) - D(\tilde{\alpha}^{(0)}) \right).
\]
Algorithm 3: DSCOVR-SAGA

- \( \bar{u}^{(0)} = Xw^{(0)} \) and \( \bar{v}^{(0)} = \frac{1}{N}X^T\alpha^{(0)} \)

- for \( t = 0, 1, 2, \ldots, T - 1 \)
  1. pick \( i \in \{1, \ldots, m\} \) and \( k \in \{1, \ldots, K\} \) with probabilities \( p_i \) and \( q_k \)

  2. compute variance-reduced stochastic gradients:

     \[
     u_i(t+1) = \bar{u}_i(t) - \frac{1}{q_k} U_{ik}(t) + \frac{1}{q_k} X_{ik} w_k(t)
     \]

     \[
     v_k(t+1) = \bar{v}_k(t) - \frac{1}{p_i} (V_{ik}(t))^T + \frac{1}{p_i} \frac{1}{N} (X_{ik})^T \alpha_i(t)
     \]

  3. update primal and dual block coordinates:

     \[
     \alpha_i^{(t+1)} = \text{prox}_{\sigma_i \psi_i^*} (\alpha_i^{(t)} + \sigma_j u_i^{(t+1)})
     \]

     \[
     w_k^{(t+1)} = \text{prox}_{\tau_k g_k} (w_k^{(t)} - \tau_k v_k^{(t+1)})
     \]

  4. update averaged historical stochastic gradients:

     \[
     \bar{u}_i(t+1) = \bar{u}_i(t) - U_{ik}(t) + X_{ik} w_k(t), \quad \bar{v}_k(t+1) = \bar{v}_k(t) - (V_{ik}(t))^T + \frac{1}{N} (X_{ik})^T \alpha_i(t)
     \]

  5. update the table of historical stochastic gradients:

     \[
     U_{ik}^{(t+1)} = X_{ik} w_k(t), \quad V_{ik}^{(t+1)} = \frac{1}{N} ((X_{ik})^T \alpha_i(t))^T
     \]

end for
Convergence of DSCOVR-SAGA

**Theorem:** Let \( \Lambda \) and \( \Gamma \) be two constants that satisfy

\[
\Lambda \geq \|X_{ik}\|^2, \quad i = 1, \ldots, m, \quad j = 1, \ldots, K,
\]

\[
\Gamma \geq \max_{i,k} \left\{ \frac{1}{p_i} \left( 1 + \frac{9m\Lambda}{2q_k N\lambda \gamma} \right), \quad \frac{1}{q_k} \left( 1 + \frac{9K\Lambda}{2p_i N\lambda \gamma} \right), \quad \frac{1}{p_i q_k} \right\}.
\]

If we choose the step sizes as

\[
\sigma_i = \frac{1}{2\gamma(p_i \Gamma - 1)}, \quad i = 1, \ldots, m,
\]

\[
\tau_k = \frac{1}{2\lambda(q_k \Gamma - 1)}, \quad k = 1, \ldots, K,
\]

then for \( t = 1, 2, \ldots, \)

\[
E \left[ \left\| \frac{w(t) - w^*}{\alpha(t) - \alpha^*} \right\|_{\lambda, \frac{\gamma}{N}}^2 \right] \leq \left( 1 - \frac{1}{3\Gamma} \right)^t \frac{4}{3} \left\| \frac{w(0) - w^*}{\alpha(0) - \alpha^*} \right\|_{\lambda, \frac{\gamma}{N}}^2
\]
**Algorithm 4: Accelerated DSCOVR**

**input:** initial points $\tilde{w}^{(0)}$, $\tilde{\alpha}^{(0)}$, and parameter $\delta > 0$

**for** $r = 0, 1, 2, \ldots$,

1. find an approximate saddle point of

$$L^{(r)}_\delta(w, a) = L(w, \alpha) + \frac{\delta \lambda}{2} \| w - \tilde{w}^{(r)} \|^2 - \frac{\delta \gamma}{2N} \| \alpha - \tilde{\alpha}^{(r)} \|^2$$

using one of the following two options:

- **option 1:** let $S = \frac{2 \log(2(1+\delta))}{\log(3/2)}$ and $T = \log(3)\Gamma_\delta$, and

  $$(\tilde{w}^{(r+1)}, \tilde{\alpha}^{(r+1)}) = \text{DSCOVR-SVRG}(\tilde{w}^{(r)}, \tilde{\alpha}^{(r)}, S, T)$$

- **option 2:** let $T = 6 \log \left(\frac{8(1+\delta)}{3}\right) \Gamma_\delta$ and

  $$(\tilde{w}^{(r+1)}, \tilde{\alpha}^{(r+1)}) = \text{DSCOVR-SAGA}(\tilde{w}^{(r)}, \tilde{\alpha}^{(r)}, T)$$

**end for**

(following techniques in Balamurugan and Bach 2016)
Convergence of accelerated DSCOVR

**Theorem:** Let \( \Lambda \) and \( \Gamma_{\delta} \) be two constants that satisfy

\[
\Lambda \geq \|X_{ik}\|^2, \quad \text{for all } i = 1, \ldots, m, \text{ and } j = 1, \ldots, K,
\]

\[
\Gamma_{\delta} \geq \max_{i,k} \left\{ \frac{1}{p_i} \left(1 + \frac{9m\Lambda}{2q_k N\lambda \gamma (1 + \delta)^2}\right), \frac{1}{q_k} \left(1 + \frac{9K\Lambda}{2p_i N\lambda \gamma (1 + \delta)^2}\right) \right\}.
\]

If we choose the step sizes as

\[
\sigma_i = \frac{1}{2\gamma (p_i \Gamma_{\delta} - 1)}, \quad i = 1, \ldots, m,
\]

\[
\tau_k = \frac{1}{2\lambda (q_k \Gamma_{\delta} - 1)}, \quad k = 1, \ldots, K,
\]

then

\[
\mathbb{E} \left[\left\| \tilde{w}^{(r)} - w^* \right\|_{\lambda, \frac{\gamma}{N}}^2 \right] \leq \left(1 - \frac{1}{2(1 + \delta)}\right)^{2r} \left\| \tilde{\alpha}^{(0)} - \alpha^* \right\|_{\lambda, \frac{\gamma}{N}}^2
\]

\]
Complexity of accelerated DSCOVR

• simplified expression for the constant \( \Gamma_\delta = K \left( 1 + \frac{9\kappa}{2(1+\delta)^2} \right) \)

• total number of block updates

\[
O \left( K \left( m(1 + \delta) + \frac{9\kappa}{2(1+\delta)} \right) \log(1 + \delta) \log \left( \frac{1}{\epsilon} \right) \right).
\]

if we choose \( \delta = \sqrt{9\kappa/(2m)} - 1 \) (assuming \( \kappa > m \)), then

\[
O \left( K \sqrt{m\kappa} \log(1 + \delta) \log \left( \frac{1}{\epsilon} \right) \right).
\]

• communication complexity (number of \( d \)-vectors transmitted):

\[
O \left( \sqrt{m\kappa} \log \left( \frac{1}{\epsilon} \right) \right)
\]

• computation complexity (number of passes over whole dataset):

\[
O \left( (1 + \sqrt{\frac{\kappa}{m}}) \log \left( \frac{1}{\epsilon} \right) \right)
\]
Implementation of DSCOVR

- C++, efficient sparse matrix operations using OpenMP
- asynchronous implementation: MPI nonblocking Send/IRecv
- also implemented Parallel GD, ADMM, CoCoA (+)
- more to come . . .
Experiments with RCV1.binary dataset

- $N = 677,399$, $d = 47236$, row normalized with $R = 1$
- run on cluster of 20 machines, 5 parameter servers, 1 master
- randomly shuffled sample and features
- smoothed hinge loss with $\ell_2$ regularization, $\lambda = 10^{-4}$
Experiments with webspam dataset

- $N = 350,000$, $d = 16,609,143$, row normalized with $R = 1$
- run on cluster of 20 workers, 10 parameter servers, 1 master
- randomly shuffled sample and features
- logistic regression with $\ell_2$ regularization, $\lambda = 10^{-4}$
DSCOVR-SAGA on webspam dataset

<table>
<thead>
<tr>
<th>nSync</th>
<th>nEpoch</th>
<th>primal_obj</th>
<th>dual_obj</th>
<th>gap</th>
<th>t_sync</th>
<th>t_comp</th>
<th>t_comm</th>
<th>t_loop</th>
<th>t_elpsdl</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.430232357706</td>
<td>0.225168873757</td>
<td>2.051e-01</td>
<td>2.025</td>
<td>0.487</td>
<td>1.453</td>
<td>2.025</td>
<td>2.025</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.361465626442</td>
<td>0.262779737691</td>
<td>9.869e-02</td>
<td>2.127</td>
<td>6.435</td>
<td>7.831</td>
<td>14.266</td>
<td>16.291</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.311349950700</td>
<td>0.278401966087</td>
<td>3.295e-02</td>
<td>2.050</td>
<td>5.685</td>
<td>8.062</td>
<td>13.747</td>
<td>30.037</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.294032397911</td>
<td>0.284556248547</td>
<td>9.476e-02</td>
<td>2.096</td>
<td>6.058</td>
<td>8.788</td>
<td>14.845</td>
<td>44.882</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.289940053505</td>
<td>0.286701605120</td>
<td>3.238e-03</td>
<td>2.024</td>
<td>5.422</td>
<td>8.101</td>
<td>13.524</td>
<td>58.406</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.288706980240</td>
<td>0.287536154538</td>
<td>1.171e-03</td>
<td>2.044</td>
<td>5.367</td>
<td>8.095</td>
<td>13.470</td>
<td>71.877</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.288254740784</td>
<td>0.287864269333</td>
<td>3.905e-04</td>
<td>2.035</td>
<td>6.212</td>
<td>8.790</td>
<td>14.993</td>
<td>86.870</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0.288128681323</td>
<td>0.287978130088</td>
<td>1.506e-04</td>
<td>2.004</td>
<td>5.569</td>
<td>8.110</td>
<td>13.680</td>
<td>100.550</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0.288088497819</td>
<td>0.288025094667</td>
<td>6.340e-05</td>
<td>2.031</td>
<td>5.436</td>
<td>8.097</td>
<td>13.532</td>
<td>114.081</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0.288073396692</td>
<td>0.288046902858</td>
<td>2.649e-05</td>
<td>2.024</td>
<td>5.364</td>
<td>8.049</td>
<td>13.422</td>
<td>127.503</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0.288068226887</td>
<td>0.288056477572</td>
<td>1.175e-05</td>
<td>2.030</td>
<td>5.364</td>
<td>8.068</td>
<td>13.421</td>
<td>140.925</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0.28806217652</td>
<td>0.288060941805</td>
<td>5.276e-06</td>
<td>2.030</td>
<td>5.336</td>
<td>8.037</td>
<td>13.378</td>
<td>154.303</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0.288065430239</td>
<td>0.288062901758</td>
<td>2.528e-06</td>
<td>2.030</td>
<td>5.334</td>
<td>8.108</td>
<td>13.437</td>
<td>167.740</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0.288065194360</td>
<td>0.288063899046</td>
<td>1.295e-06</td>
<td>2.024</td>
<td>5.337</td>
<td>8.028</td>
<td>13.364</td>
<td>181.104</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0.288065015129</td>
<td>0.288064394949</td>
<td>6.202e-07</td>
<td>2.029</td>
<td>5.318</td>
<td>8.064</td>
<td>13.403</td>
<td>194.507</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0.288064917447</td>
<td>0.288064625062</td>
<td>2.924e-07</td>
<td>2.026</td>
<td>5.353</td>
<td>8.003</td>
<td>13.357</td>
<td>207.864</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0.288064853838</td>
<td>0.288064735092</td>
<td>1.503e-07</td>
<td>2.030</td>
<td>5.387</td>
<td>8.073</td>
<td>13.439</td>
<td>221.302</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0.288064867950</td>
<td>0.288064791393</td>
<td>7.656e-08</td>
<td>2.039</td>
<td>5.625</td>
<td>8.078</td>
<td>13.704</td>
<td>235.006</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0.288064852335</td>
<td>0.288064821789</td>
<td>3.055e-08</td>
<td>2.023</td>
<td>6.698</td>
<td>9.328</td>
<td>16.023</td>
<td>251.029</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0.288064850799</td>
<td>0.288064834053</td>
<td>1.675e-08</td>
<td>2.031</td>
<td>5.736</td>
<td>8.052</td>
<td>13.790</td>
<td>264.820</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0.288064848282</td>
<td>0.288064840064</td>
<td>8.218e-09</td>
<td>2.003</td>
<td>6.378</td>
<td>8.633</td>
<td>15.025</td>
<td>279.845</td>
</tr>
</tbody>
</table>
The cost of synchronization

The plots show the objective gap and communication time for different algorithms over time and the number of passes over data.

- **Objective Gap**
  - ADMM
  - ParaGD
  - CoCoA+
  - DSCOVR
  - The graphs demonstrate how each algorithm converges to the optimal solution over time.

- **Communication Time**
  - The communication time is plotted against the number of iterations or passes over data.
  - Different algorithms have varying levels of communication overhead, which can affect their performance.

These graphs help in understanding the trade-offs between computation and communication in distributed optimization algorithms.
Summary

DSCOVR

• saddle-point formulation allows simultaneous partition of both data and model to gain parallelism
• used stochastic variance reduction to achieve fast convergence
• asynchronous, event-driven implementation
• no simultaneous updates, no stale states of delays to worry
• improved computation complexity for distributed ERM

additional features

• DSCOVR-SVRG only need to communicate sparse vectors
• also developed dual-free version of primal-dual algorithms (using technique from Lan 2015)