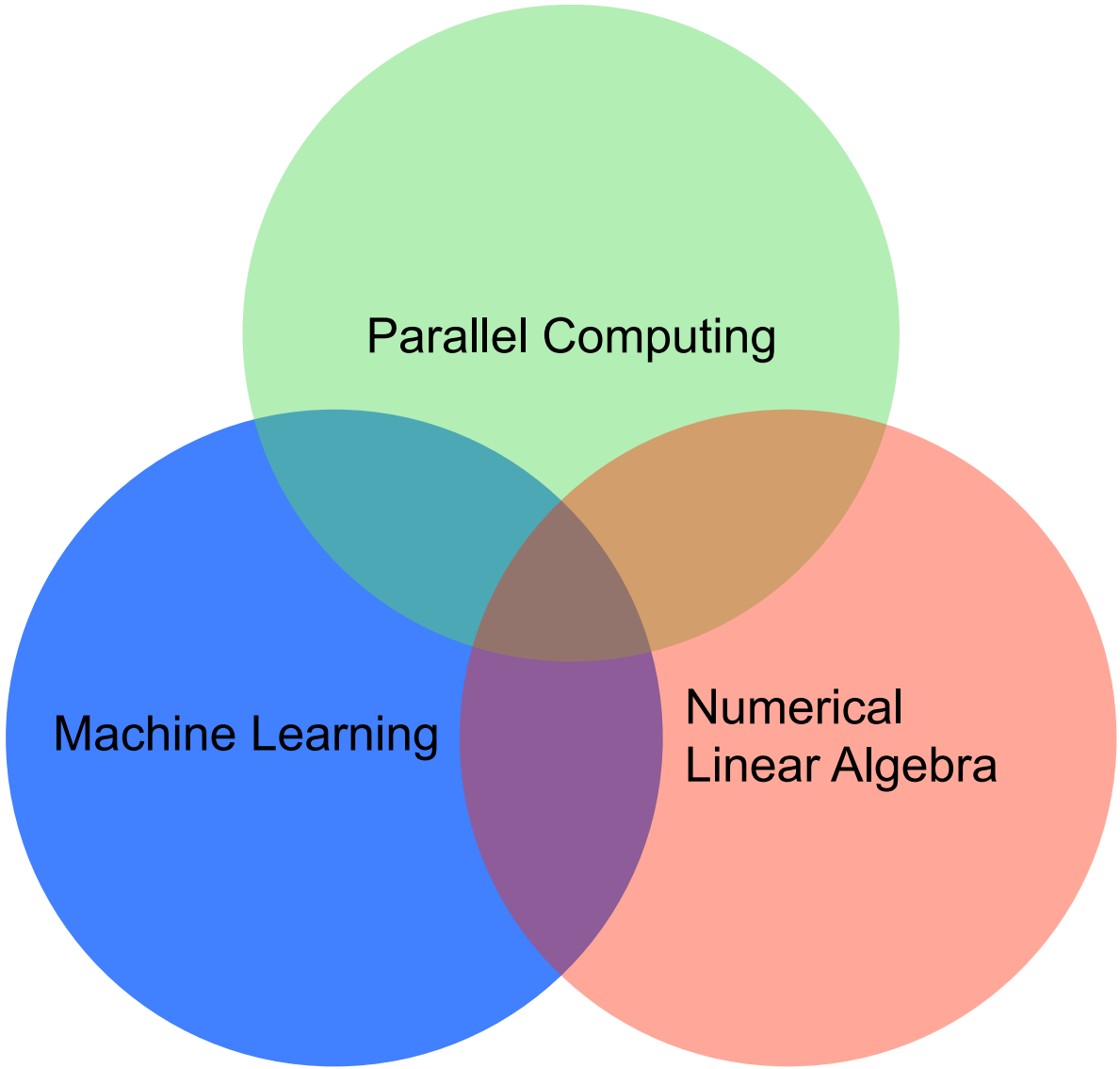
Communication-avoiding coordinate descent methods for linear systems

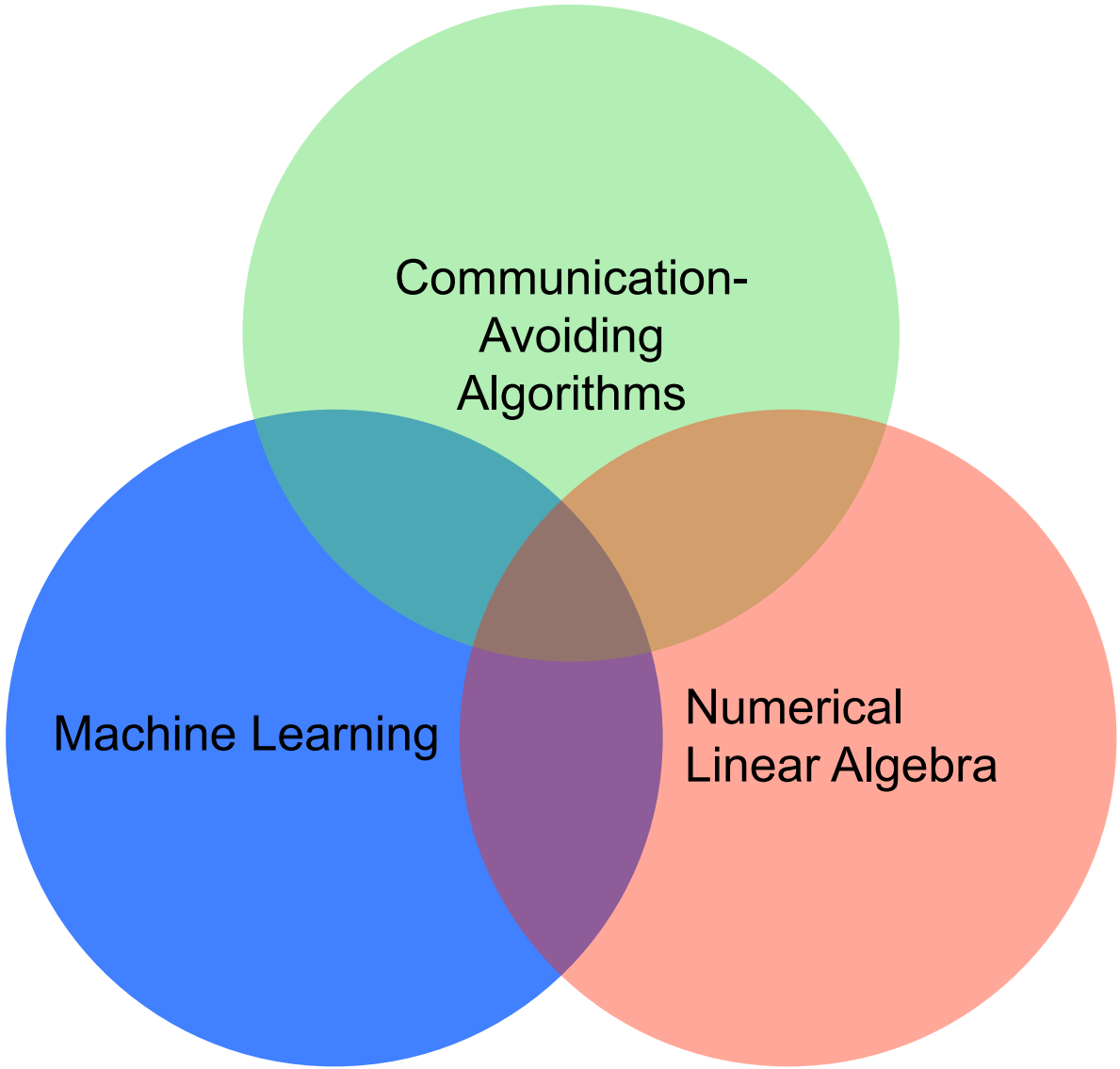
Aditya Devarakonda
with: Kimon Fountoulakis, Jim Demmel and Michael Mahoney

UC Berkeley

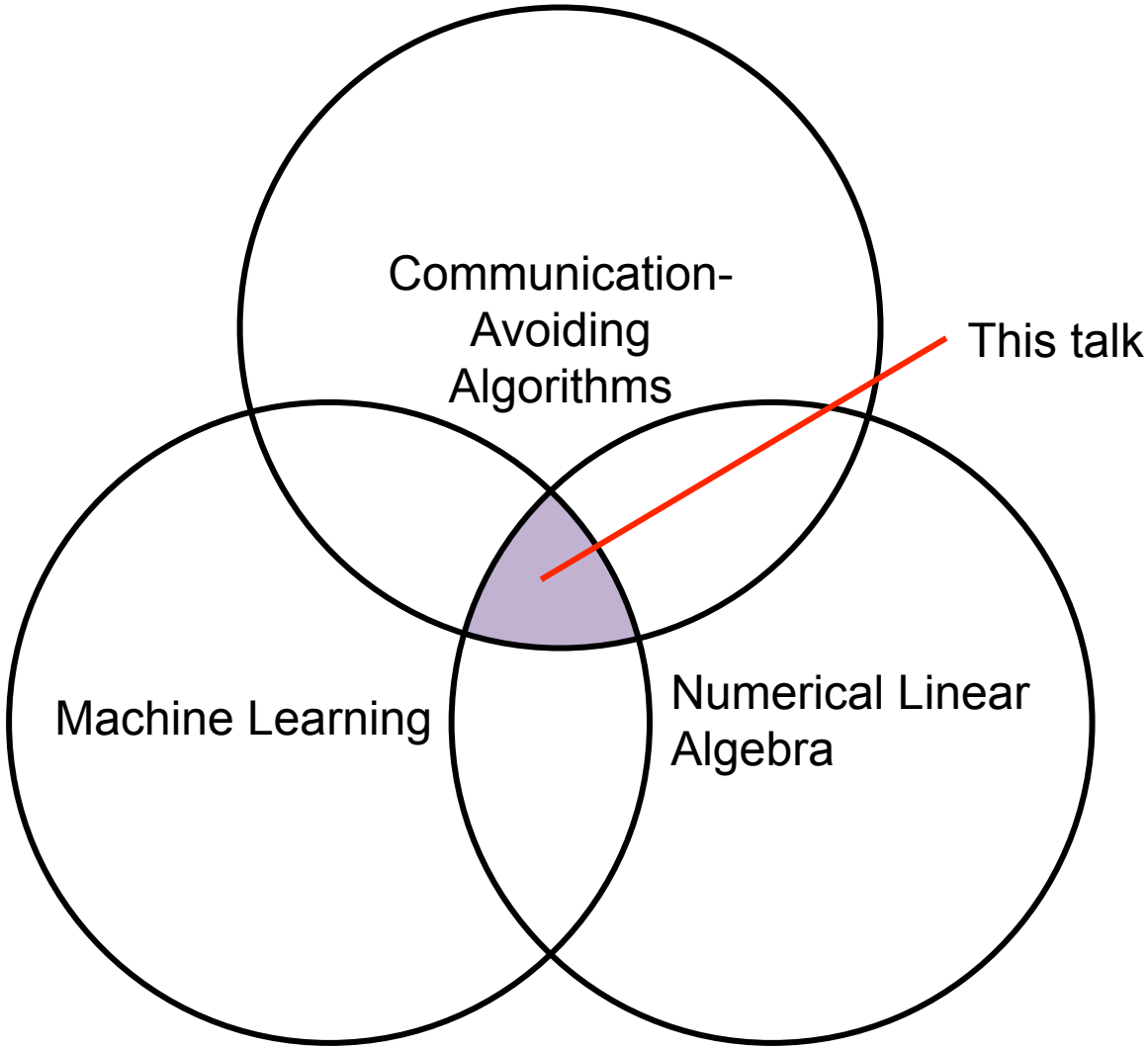
Introduction



Introduction



Introduction



Preliminaries: Cost Model

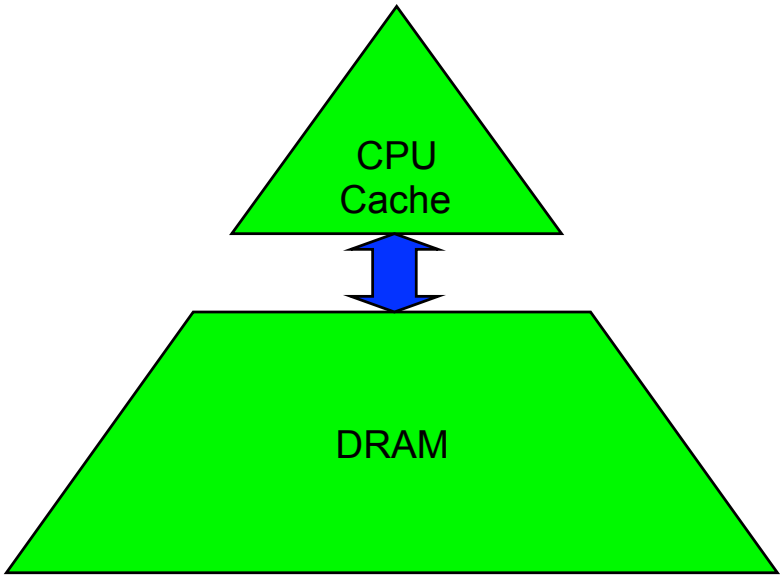


Figure 1: Sequential Case

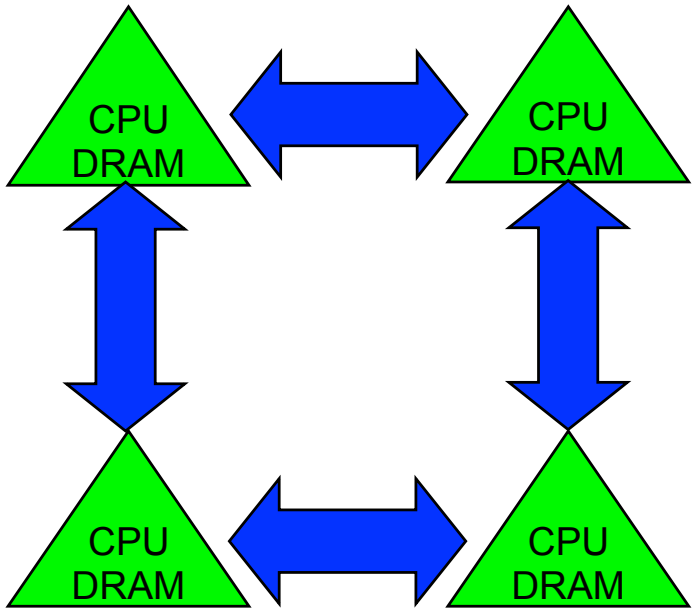


Figure 2: Parallel Case

Algorithms have two costs: **Computation** and **Communication.**

Preliminaries: Cost Model

Computation: flops

Communication: bandwidth
latency

Running Time Model:
$(\gamma \times \text{flops}) + (\beta \times \text{amounts of data moved}) + (\alpha \times \#\text{ of messages})$.

$\gamma =$ time per floating point operation
$\beta =$ time per unit of data moved
$\alpha =$ time per message

Preliminaries: Cost Model

$$\gamma << \beta << \alpha$$

Gap is growing.

Preliminaries: Cost Model

$$\gamma << \beta << \alpha$$

Gap is growing.

Avoid communication to save time.

Preliminaries: Cost Model

$$\gamma << \frac{1}{\beta} << \alpha$$

Gap is growing.

Avoid communication to save time and energy.

Preliminaries: Communication Costs

Processors can communicate several ways:

Preliminaries: Communication Costs

Processors can communicate several ways:

point to point.

Preliminaries: Communication Costs

Processors can communicate several ways:

- point to point.
- one to many.

Preliminaries: Communication Costs

Processors can communicate several ways:

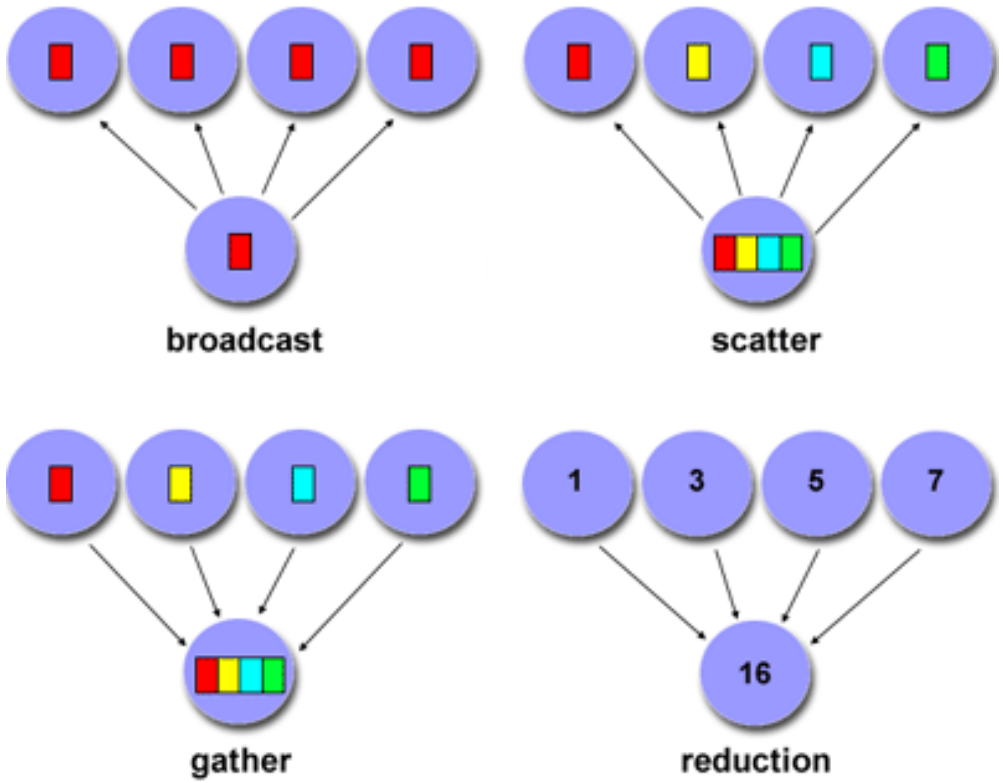
- point to point.
- one to many.
- many to one.

Preliminaries: Communication Costs

Processors can communicate several ways:

- point to point.
- one to many.
- many to one.
- all to all.

Preliminaries: Communication Costs



Preliminaries: Communication Costs

Communication Primitive	Routing Algorithm	Latency (L)	Bandwidth (W)
Point-to-Point	–	$O(1)$	$O(d)$
Broadcast, Reduction and Scatter	Virtual N-ary Tree	$O(\log P)$	$O(d \log P)$
Gather	Virtual N-ary Tree	$O(\log P)$	$O(dP \log P)$

$P = \#$ of processors

$d =$ amount of data moved

Preliminaries: Communication Costs

Communication Primitive	Routing Algorithm	Latency (L)	Words (W)
Point-to-Point	–	$O(1)$	$O(d)$
Broadcast, Reduction and Scatter	Virtual N-ary Tree	$O(\log P)$	$O(d \log P)$
Gather	Virtual N-ary Tree	$O(\log P)$	$O(dP \log P)$
All-to-All	Recursive Doubling (Short Message)	$O(\log P)$	$O(dP \log P)$
	Virtual Ring (Long Message)	$O(P)$	$O(dP)$

Preliminaries: CA-Krylov Methods

Basic kernel is a matrix-vector product (no CA version).

Preliminaries: CA-Krylov Methods

Basic kernel is a matrix-vector product (no CA version).

Strategy: Unroll Krylov method iteration loop and apply matrix powers kernel.

Factor of s savings in communication (Hoemmen, Carson, Demmel and others).

Regularized Least Squares

Given $X \in \mathbb{R}^{m \times n}$ $(m < n)$ and $y \in \mathbb{R}^n$, find $w \in \mathbb{R}^m$ such that

$$\min_{w \in \mathbb{R}^m} \frac{1}{2n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \|w\|_2^2$$

Closed form solution: $w = \left(XX^T + \lambda n I_m\right)^{-1} Xy$.

Regularized Least Squares

Many ways to solve using direct and iterative methods.

Regularized Least Squares

Many ways to solve using direct and iterative methods.

We solve it using stochastic dual coordinate ascent (SDCA).

Comparable to (or better than) stochastic gradient descent (SGD).

SDCA derivation

Primal problem:

$$\min_{w \in \mathbb{R}^m} \frac{1}{2n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \|w\|_2^2$$

Dual problem[1]:

$$\max_{\alpha \in \mathbb{R}^n} -\frac{1}{2n} \sum_{i=1}^{n} \alpha_i^2 + \alpha_i y_i - \frac{\lambda}{2} \left\| \frac{1}{\lambda n} X \alpha \right\|_2^2$$

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{2n} \|\alpha + y\|^2 + \frac{\lambda}{2} \left\| \frac{1}{\lambda n} X \alpha \right\|_2^2 .$$

[1] By definition of conjugate functions (Legendre Transform)

SDCA derivation

Primal-Dual Relationship:

$$w = -\frac{1}{\lambda n} X \alpha$$

SDCA derivation

Dual problem (not an algorithm):

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{2n} \|\alpha + y\|^2 + \frac{\lambda}{2} \left\| \frac{1}{\lambda n} X\alpha \right\|_2^2 .$$

SDCA derivation

Dual problem (not an algorithm):

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{2n} \|\alpha + y\|^2 + \frac{\lambda}{2} \left\| \frac{1}{\lambda n} X\alpha \right\|_2^2 .$$

Coordinate-wise update method:

Let h be an iteration index.
Select a coordinate $i \in [n]$.

$$\min_{\Delta\alpha \in \mathbb{R}} \frac{1}{2n} \left\| \alpha^{(h-1)} + \Delta\alpha e_i + y \right\|_2^2 + \frac{\lambda}{2} \left\| \frac{1}{\lambda n} X\alpha^{(h-1)} + \frac{\Delta\alpha}{\lambda n} x_i \right\|^2$$

$$\alpha^{(h)} = \alpha^{(h-1)} + \Delta\alpha e_i .$$

SDCA derivation

$$\min_{\Delta\alpha\in\mathbb{R}} \frac{1}{2n}\left\|\alpha^{(h-1)}+\Delta\alpha e_i+y\right\|_2^2+\frac{\lambda}{2}\left\|\frac{1}{\lambda n}X\alpha^{(h-1)}+\frac{\Delta\alpha}{\lambda n}x_i\right\|^2$$

Solve for $\Delta\alpha$:

$$\Delta\alpha=\frac{-1}{\frac{1}{\lambda n^2}x_i^Tx_i+\frac{1}{n}}e_i^T\left\{\left(\frac{1}{\lambda n^2}X^TX+\frac{1}{n}I\right)\alpha^{(h-1)}+\frac{1}{n}y\right\}.$$

SDCA derivation

$$\Delta\alpha = \frac{-1}{\frac{1}{\lambda n^2} x_i^T x_i + \frac{1}{n}} e_i^T \left\{ \left(\frac{1}{\lambda n^2} X^T X + \frac{1}{n} I \right) \alpha^{(h-1)} + \frac{1}{n} y \right\}.$$

Let[2]:

$$\gamma_i = \frac{-1}{\frac{1}{\lambda n^2} x_i^T x_i + \frac{1}{n}}$$

$$A = \frac{1}{\lambda n^2} X^T X + \frac{1}{n} I$$

$$r^{(h-1)} = A\alpha^{(h-1)} + \frac{1}{n} y.$$

[2] A is a notational change, not explicitly formed

SDCA derivation

Before:

$$\Delta\alpha = \frac{-1}{\frac{1}{\lambda n^2} x_i^T x_i + \frac{1}{n}} e_i^T \left\{ \left(\frac{1}{\lambda n^2} X^T X + \frac{1}{n} I \right) \alpha^{(h-1)} + \frac{1}{n} y \right\}$$

$$\alpha^{(h)} = \alpha^{(h-1)} + \Delta\alpha e_i$$

$$w^{(h)} = w^{(h-1)} - \frac{1}{\lambda n} \Delta\alpha X e_i.$$

After:

$$\Delta\alpha = \gamma_i e_i^T r^{(h-1)}$$

$$\alpha^{(h)} = \alpha^{(h-1)} + \gamma_i e_i e_i^T r^{(h-1)}$$

$$w^{(h)} = w^{(h-1)} - \frac{1}{\lambda n} \Delta\alpha X e_i.$$

SDCA residual recurrence

Since $r^{(h-1)} = A\alpha^{(h-1)} + \frac{1}{n}y$
and $\alpha^{(h)} = \alpha^{(h-1)} + \gamma_i e_i e_i^T r^{(h-1)}$

$r^{(h)} = \left(I + \gamma_i A e_i e_i^T\right) r^{(h-1)}$

This residual update helps us avoid communication.

SDCA Krylov-like Basis

Let $i_j \in [n]$ (an index chosen from 1 to n) and $r^{(0)}$ be the starting residual.

$$\mathcal{R}_1 := \Bigg[r^{(0)}, \left(I + \gamma_{i_1} A e_{i_1} e_{i_1}^T\right) r^{(0)}, \left(I + \gamma_{i_2} A e_{i_2} e_{i_2}^T\right)\left(I + \gamma_{i_1} A e_{i_1} e_{i_1}^T\right) r^{(0)}, \ldots, \prod_{j=s-1}^{1} \left(I + \gamma_{i_j} A e_{i_j} e_{i_j}^T\right) r^{(0)} \Bigg] \in \mathbb{R}^{n \times s}.$$

This "unrolls" the residual by s iterations.

SDCA Krylov-like Basis

Let $i_j \in [n]$ (an index chosen from 1 to n) and $r^{(0)}$ be the starting residual.

$$\mathcal{A}_{sh} := \Bigg[\gamma_{i_1} e_{i_1}^T r^{(0)}, \gamma_{i_2} e_{i_2}^T \left(I + \gamma_{i_1} A e_{i_1} e_{i_1}^T \right) r^{(0)}, \\ \gamma_{i_3} e_{i_3}^T \left(I + \gamma_{i_2} A e_{i_2} e_{i_2}^T \right) \left(I + \gamma_{i_1} A e_{i_1} e_{i_1}^T \right) r^{(0)}, \\ \ldots, \gamma_{i_s} e_{i_s}^T \prod_{j=s-1}^{1} \left(I + \gamma_{i_j} A e_{i_j} e_{i_j}^T \right) r^{(0)} \Bigg] \in \mathbb{R}^{1 \times s}.$$

Due to unrolling the residual.

SDCA Krylov-like Basis

The hard part is computing: $\gamma_{i_s} e_{i_s}^T \prod_{j=s-1}^{1} \left(I + \gamma_{i_j} A e_{i_j} e_{i_j}^T \right) r^{(0)}$

However, recall that

$\gamma_{i_j} = \frac{-1}{\frac{1}{\lambda n^2} x_{i_j}^T x_{i_j} + \frac{1}{n}}$ requires a dot-product.

SDCA Krylov-like Basis

The hard part is computing: $\gamma_{i_s} e_{i_s}^T \prod_{j=s-1}^{1} \left(I + \gamma_{i_j} A e_{i_j} e_{i_j}^T \right) r^{(0)}$

However, recall that

$\gamma_{i_j} = \frac{-1}{\frac{1}{\lambda n^2} x_{i_j}^T x_{i_j} + \frac{1}{n}}$ requires a dot-product.

Easy to avoid communication. Requires one reduction.

SDCA Krylov-like Basis

Recall that: $A := \frac{1}{\lambda n^2} X^T X + \frac{1}{n} I$

Notice the $e_{i_j}^T$ terms: $\gamma_{i_s} e_{i_s}^T \prod_{j=s-1}^{1} \left(I + \gamma_{i_j} A e_{i_j} e_{i_j}^T \right) r^{(0)}$

Expanding the product results in dot-products.

SDCA Krylov-like Basis

Let $Y = [x_{i_1}, x_{i_2}, \ldots x_{i_s}] \in \mathbb{R}^{m \times s}$
then compute the Gram-like matrix $G = \frac{1}{\lambda n^2} Y^T Y + \frac{1}{n} I$.

SDCA Krylov-like Basis

Let $Y = [x_{i_1}, x_{i_2}, \ldots x_{i_s}] \in \mathbb{R}^{m \times s}$
then compute the Gram-like matrix $G = \frac{1}{\lambda n^2} Y^T Y + \frac{1}{n} I$.

G contains all of the coefficients required to compute $\mathcal{A}_1$.

SDCA Krylov-like Basis

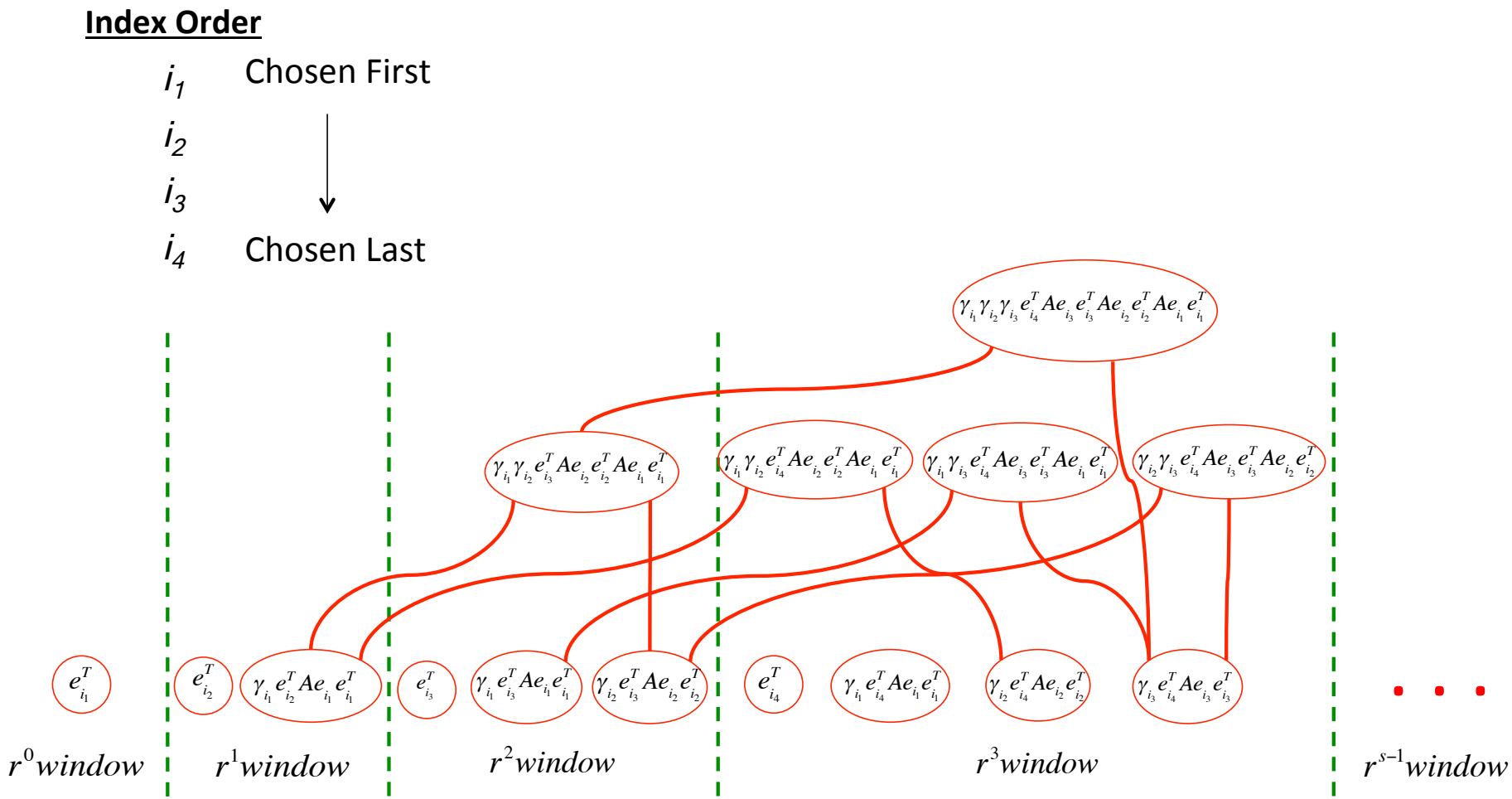
Let $Y = [x_{i_1}, x_{i_2}, \ldots x_{i_s}] \in \mathbb{R}^{m \times s}$
then compute the Gram-like matrix $G = \frac{1}{\lambda n^2} Y^T Y + \frac{1}{n} I$.

G contains all of the coefficients required to compute $\mathcal{A}_1$.

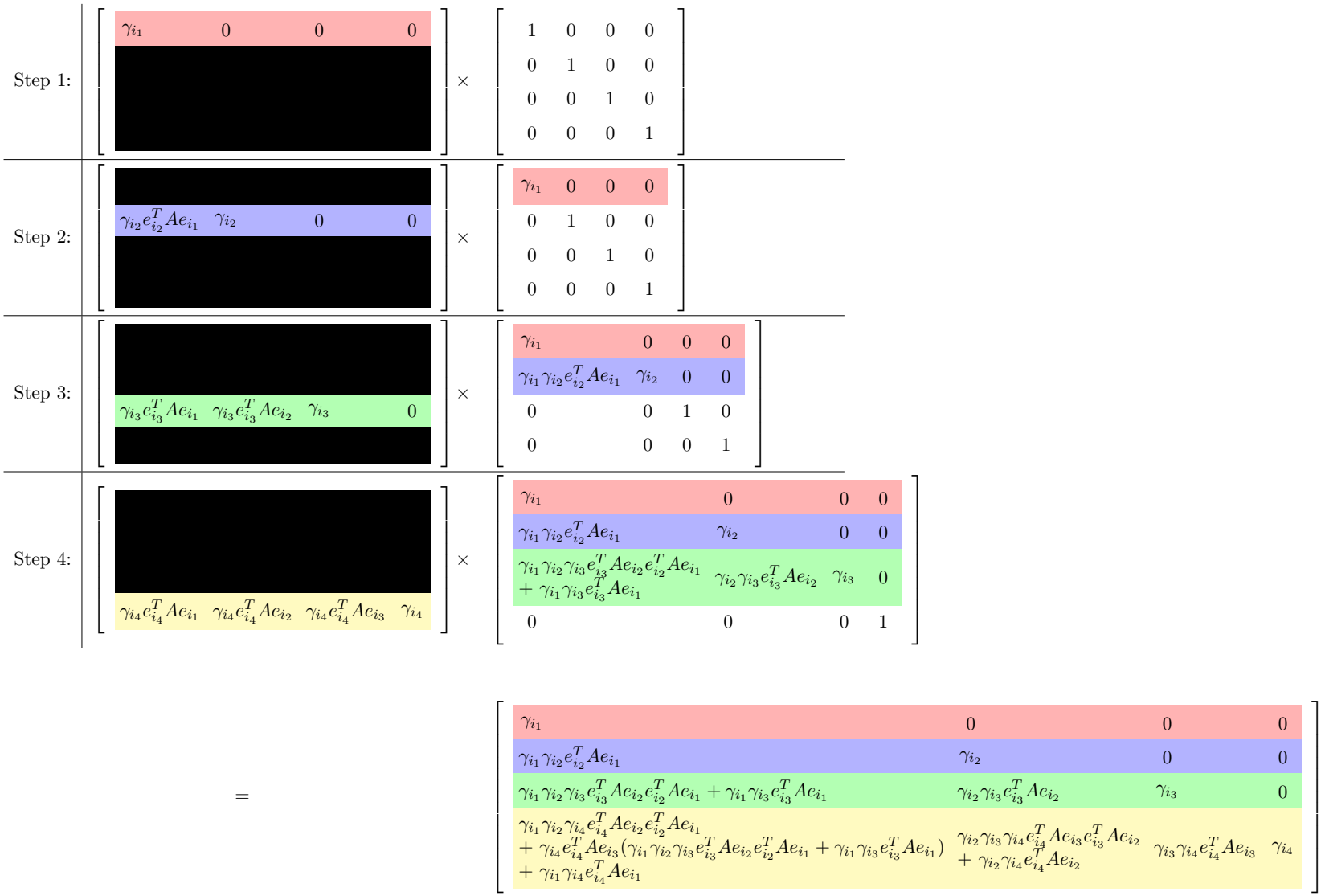
Computing G requires a single reduction.

CA-SDCA Dependency Graph

G contains the coefficients but still need to compute the expansion.



SDCA Polynomial Expansion



CA-SDCA Step Sizes

$$\begin{bmatrix} \Delta\alpha_{i_1} \\ \Delta\alpha_{i_2} \\ \Delta\alpha_{i_3} \\ \Delta\alpha_{i_4} \end{bmatrix} = \begin{bmatrix} \gamma_{i_1} & 0 & 0 & 0 \\ \gamma_{i_1}\gamma_{i_2} e_{i_2}^T A e_{i_1} & \gamma_{i_2} & 0 & 0 \\ \gamma_{i_1}\gamma_{i_2}\gamma_{i_3} e_{i_3}^T A e_{i_2} e_{i_2}^T A e_{i_1} + \gamma_{i_1}\gamma_{i_3} e_{i_3}^T A e_{i_1} & \gamma_{i_2}\gamma_{i_3} e_{i_3}^T A e_{i_2} & \gamma_{i_3} & 0 \\ \gamma_{i_1}\gamma_{i_2}\gamma_{i_4} e_{i_4}^T A e_{i_2} e_{i_2}^T A e_{i_1} \\ + \gamma_{i_4} e_{i_4}^T A e_{i_3} (\gamma_{i_1}\gamma_{i_2}\gamma_{i_3} e_{i_3}^T A e_{i_2} e_{i_2}^T A e_{i_1} + \gamma_{i_1}\gamma_{i_3} e_{i_3}^T A e_{i_1}) \\ + \gamma_{i_1}\gamma_{i_4} e_{i_4}^T A e_{i_1} & \gamma_{i_2}\gamma_{i_3}\gamma_{i_4} e_{i_4}^T A e_{i_3} e_{i_3}^T A e_{i_2} \\ & + \gamma_{i_2}\gamma_{i_4} e_{i_4}^T A e_{i_2} & \gamma_{i_3}\gamma_{i_4} e_{i_4}^T A e_{i_3} & \gamma_{i_4} \end{bmatrix} \times \begin{bmatrix} r_{i_1}^{(0)} \\ r_{i_2}^{(0)} \\ r_{i_3}^{(0)} \\ r_{i_4}^{(0)} \end{bmatrix}$$

No communication required.

CA-SDCA Communication Cost

The subroutine packs s iterations into a Gram-matrix and matrix-vector computations.

Algorithm	Latency (L)	Bandwidth (W)
SDCA	$O(H \log P)$	$O(Hm \log P)$
CA-SDCA	$O(\frac{H}{s} \log P)$	$O(Hs \log P + \frac{Hm}{s} \log P)$

Reduces latency cost.
Reduces bandwidth cost if $s^2 < m$.

CA-SDCA Results

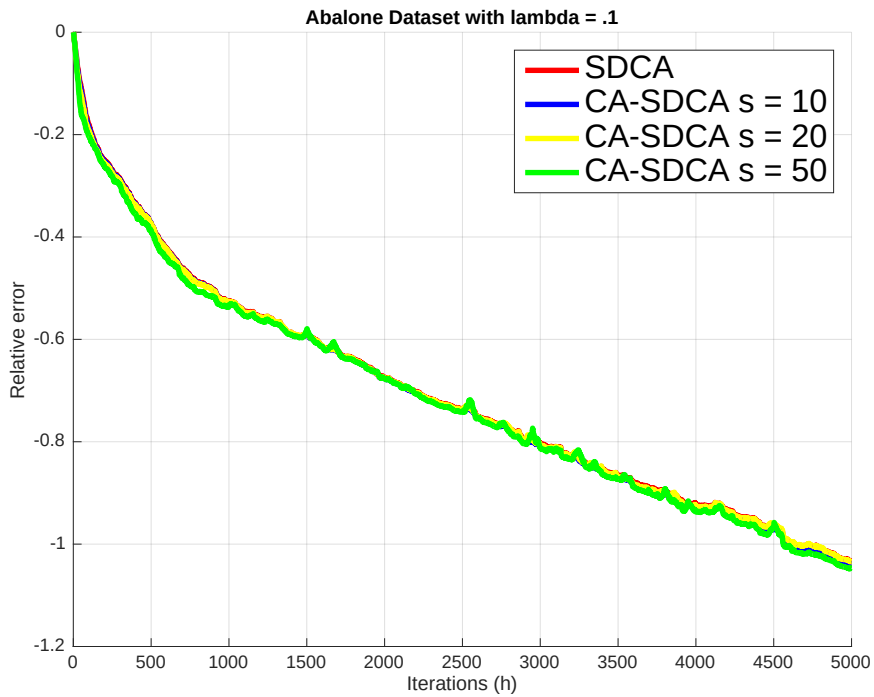
Matlab experiments using datasets from libsvm.

Measuring relative error: $\frac{\|w_{alg} - w_{opt}\|}{\|w_{opt}\|}$

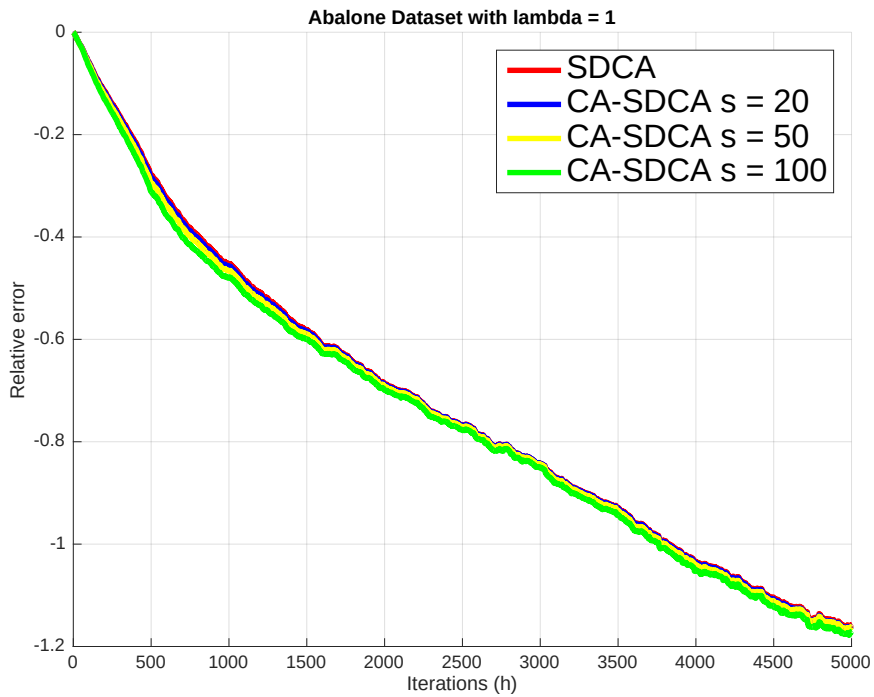
w_{opt} obtained from conjugate gradients.

Dataset name	Dimensions	Problem type
Abalone	4177×8	regression
w1a	2477×300	classification
a9a	32561×123	classification

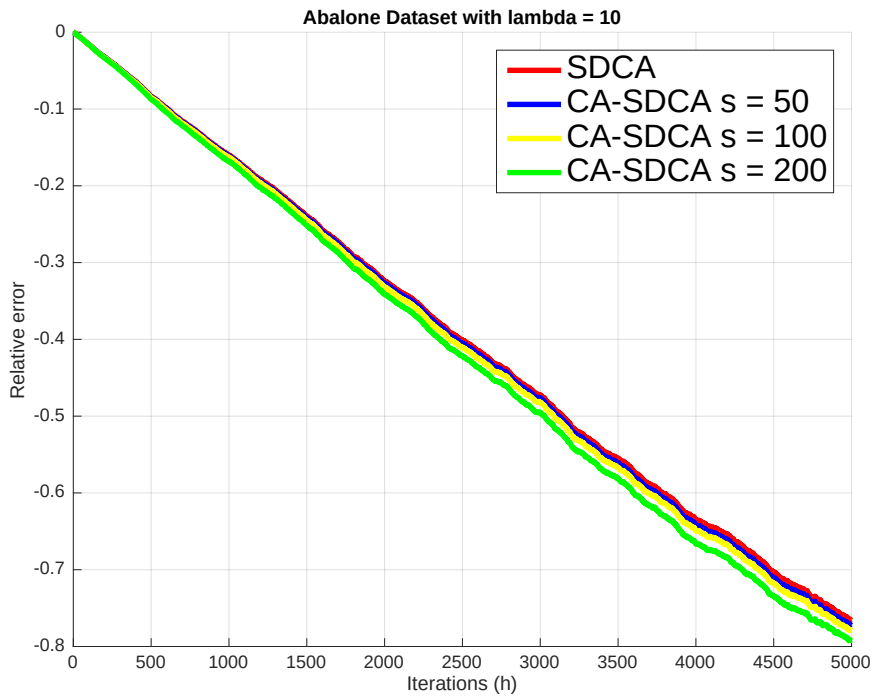
CA-SDCA Results: Abalone



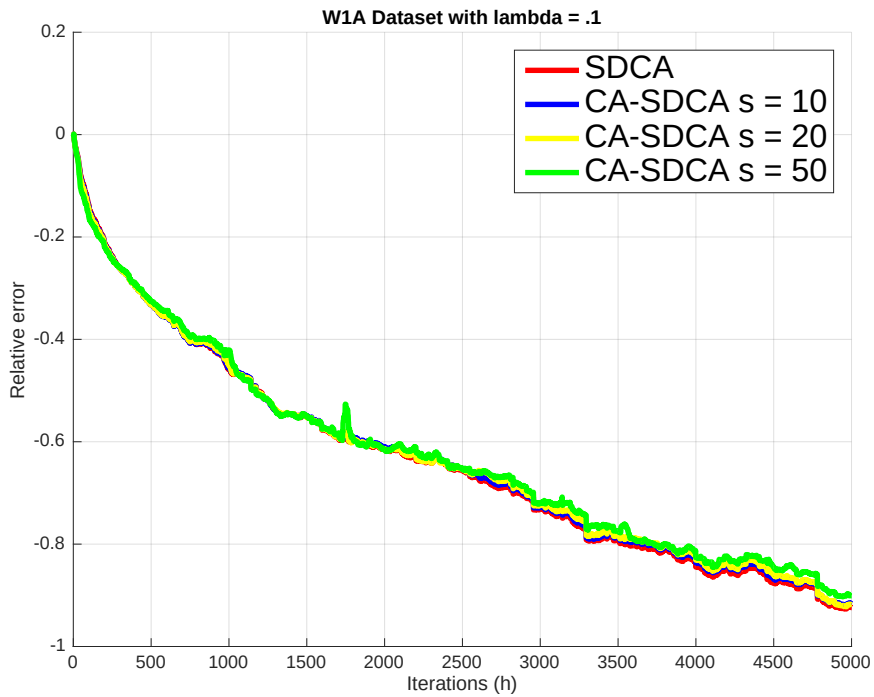
CA-SDCA Results: Abalone



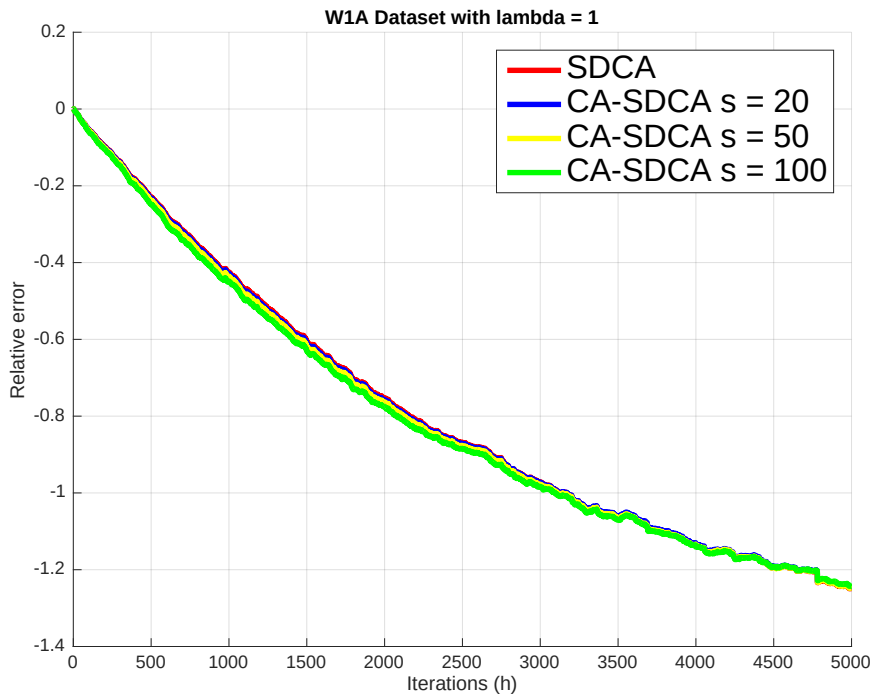
CA-SDCA Results: Abalone



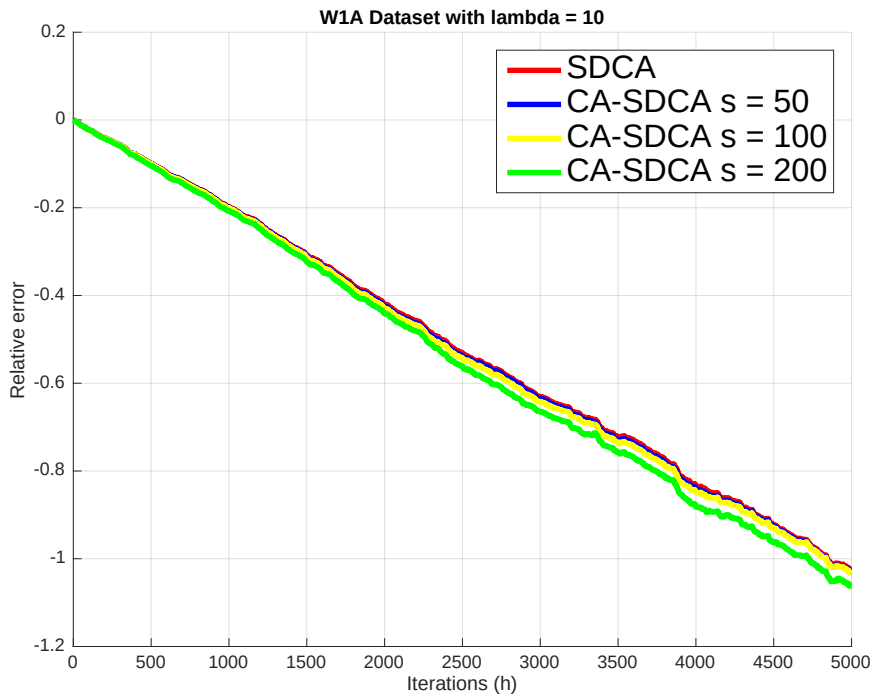
CA-SDCA Results: w1a



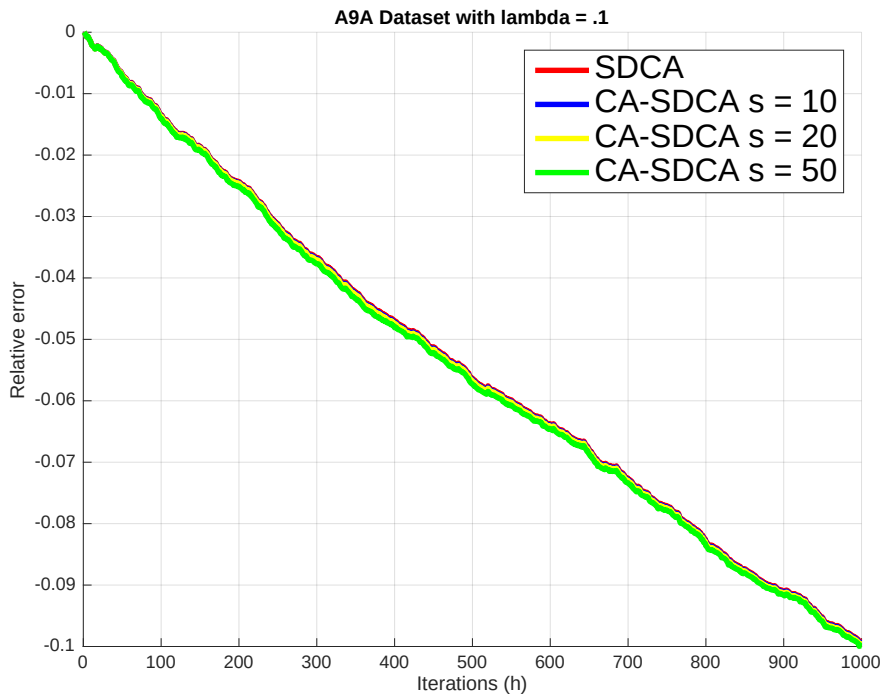
CA-SDCA Results: w1a



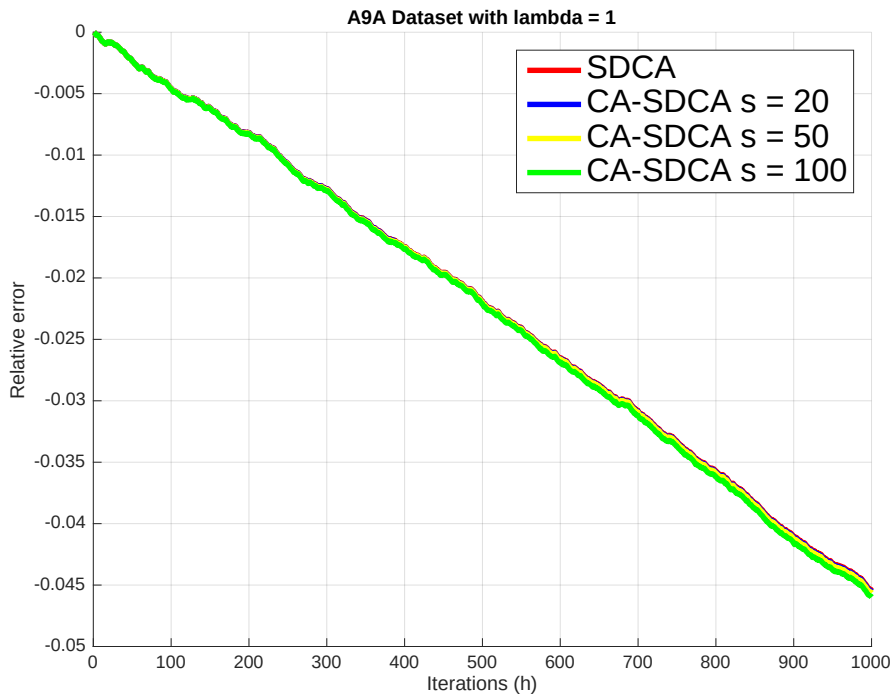
CA-SDCA Results: w1a



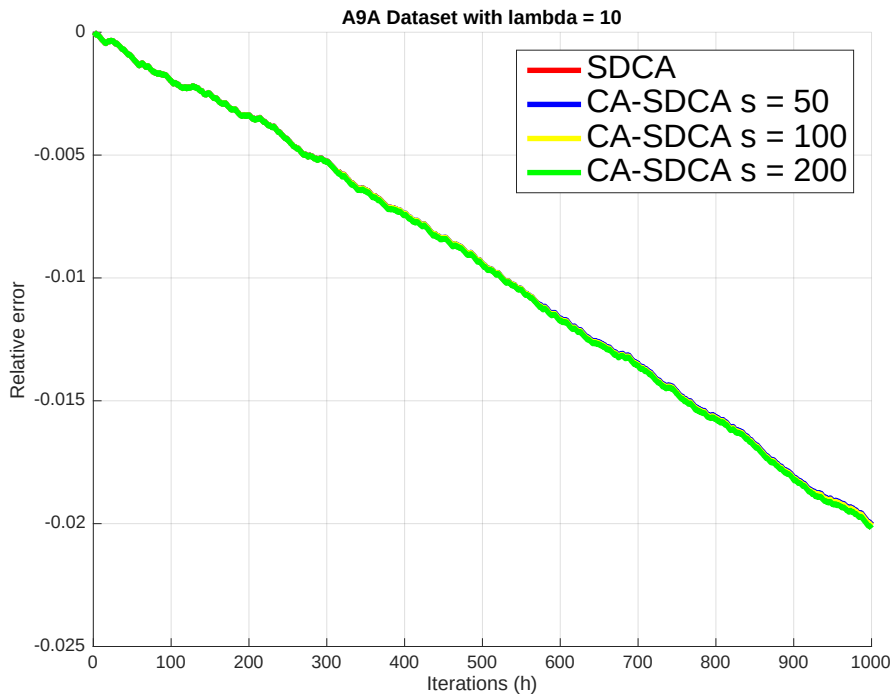
CA-SDCA Results: a9a



CA-SDCA Results: a9a



CA-SDCA Results: a9a



CA-SDCA Summary

Maximum attainable s depends on conditioning of X and λ.

CA-SDCA Summary

Maximum attainable s depends on conditioning of X and λ.

In practice, s should be chosen to balance:

- numerical error
- computation vs. communication

CA-mini-batch SDCA Derivation

Updates batches of b coordinates independently.

CA-mini-batch SDCA Derivation

Recall for SDCA:

$$r^{(h)} = \left(I + \gamma_i A e_i e_i^T\right) r^{(h-1)}$$

residual update for mbSDCA:

$$r^{(h)} = \left(I + \sum_{k=1}^{b} \gamma_{i_k} A e_{i_k} e_{i_k}^T\right) r^{(h-1)}$$

CA-mini-batch SDCA Derivation

Let:

$$\mathbb{I}_h = \begin{bmatrix} e_{i_1} & e_{i_2} & \cdots & e_{i_k} & \cdots & e_{i_b} \end{bmatrix},$$

and define the diagonal matrix $\Gamma_h \in \mathbb{R}^{b \times b}$ such that $\Gamma_h (i_k, i_k) = \gamma_{i_k}$.

then residual update for mbSDCA:

$$r^{(h)} = \left(I + A \mathbb{I}_h \Gamma_h \mathbb{I}_h^T \right) r^{(h-1)}.$$

CA-mini-batch SDCA Derivation

All updates are now block updates:

$$\mathbb{A}_h = \begin{bmatrix} \Delta\alpha_{i_1} \\ \vdots \\ \Delta\alpha_{i_k} \\ \vdots \\ \Delta\alpha_{i_b} \end{bmatrix} = \Gamma_h \mathbb{I}_h^T r^{(h-1)}$$

$$\alpha^{(h)} = \alpha^{(h-1)} + \mathbb{I}_h \mathbb{A}_h$$

$$w^{(h)} = w^{(h-1)} - \frac{1}{\lambda n} X \mathbb{I}_h \mathbb{A}_h.$$

CA-mini-batch SDCA Krylov-like Basis

$$\mathcal{R}_1 := \Bigg[r^{(0)}, \left(I + A\mathbb{I}_1\Gamma_1\mathbb{I}_1^T \right) r^{(0)}, \left(I + A\mathbb{I}_2\Gamma_2\mathbb{I}_2^T \right)\left(I + A\mathbb{I}_1\Gamma_1\mathbb{I}_1^T \right) r^{(0)}, \ldots, \prod_{j=s-1}^{1} \left(I + A\mathbb{I}_j\Gamma_j\mathbb{I}_j^T \right) r^{(0)} \Bigg] \in \mathbb{R}^{n\times s}.$$

CA-mini-batch SDCA Krylov-like Basis

$$\mathcal{A}_1 := \Big[\Gamma_1 \mathbb{I}_1^T r^{(0)}, \Gamma_2 \mathbb{I}_2^T \left(I + A \mathbb{I}_1 \Gamma_1 \mathbb{I}_1^T \right) r^{(0)}, \\ \Gamma_3 \mathbb{I}_3^T \left(I + \mathbb{I}_2 A \Gamma_2 \mathbb{I}_2^T \right) \left(I + A \mathbb{I}_1 \Gamma_1 \mathbb{I}_1^T \right) r^{(0)}, \\ \ldots, \Gamma_s \mathbb{I}_s^T \prod_{j=s-1}^{1} \left(I + A \mathbb{I}_j \Gamma_j \mathbb{I}_j^T \right) r^{(0)} \Big] \in \mathbb{R}^{b \times s}.$$

CA-mini-batch SDCA Dependency Graph

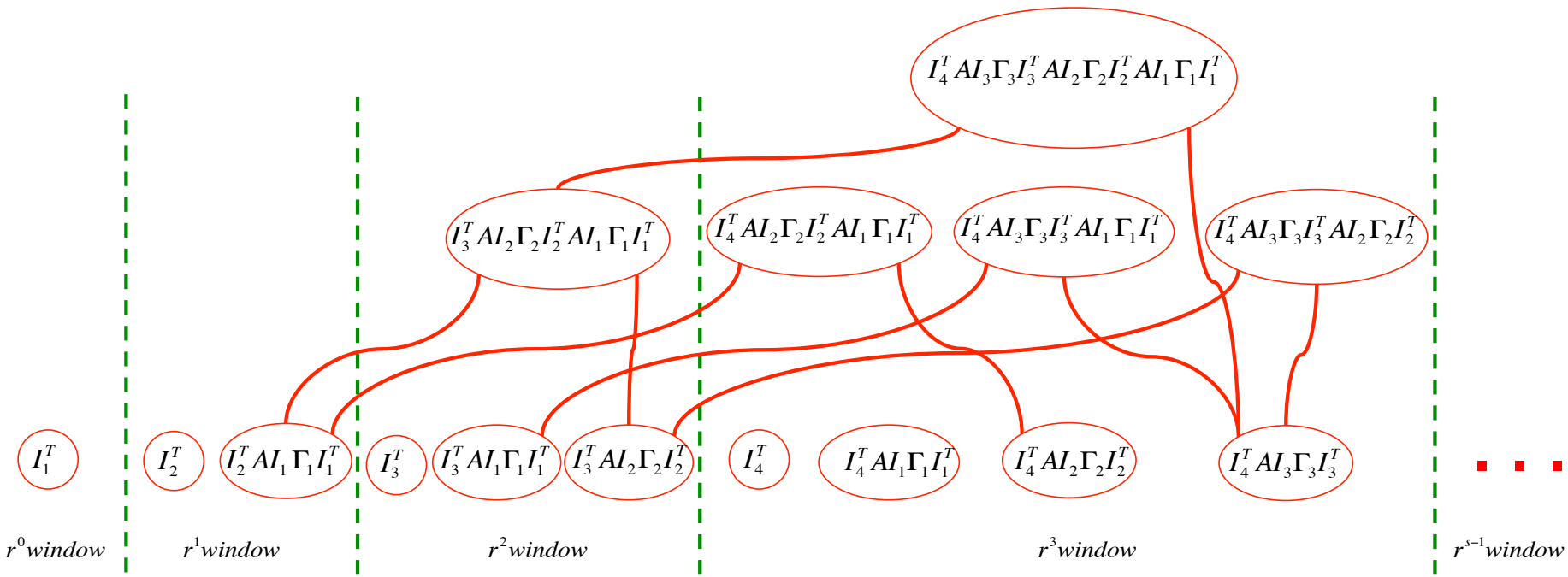
Index Order

I_1 Chosen First

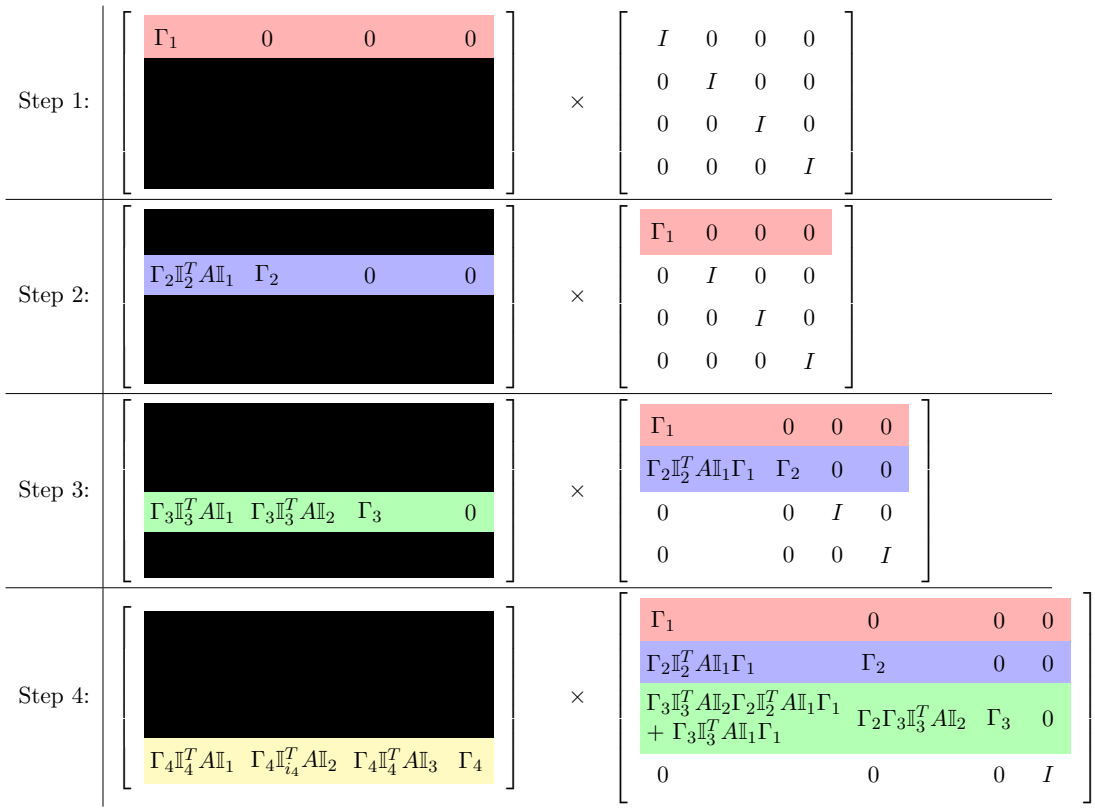
I_2 ↓

I_3

I_4 Chosen Last



CA-mini-batch SDCA Poly. Expansion



=

$$\begin{bmatrix}
\Gamma_1 & 0 & 0 & 0 \\
\Gamma_2\mathbb{I}_2^T A\mathbb{I}_1\Gamma_1 & \Gamma_2 & 0 & 0 \\
\Gamma_3\mathbb{I}_3^T A\mathbb{I}_2\Gamma_2\mathbb{I}_2^T A\mathbb{I}_1\Gamma_1 + \Gamma_3\mathbb{I}_3^T A\mathbb{I}_1\Gamma_1 & \Gamma_2\Gamma_3\mathbb{I}_3^T A\mathbb{I}_2 & \Gamma_3 & 0 \\
\Gamma_4\mathbb{I}_4^T A\mathbb{I}_1\Gamma_1 + \Gamma_4\mathbb{I}_4^T A\mathbb{I}_3(\Gamma_3\mathbb{I}_3^T A\mathbb{I}_2\Gamma_2\mathbb{I}_2^T A\mathbb{I}_1\Gamma_1 + \Gamma_3\mathbb{I}_3^T A\mathbb{I}_1\Gamma_1) + \Gamma_4\mathbb{I}_4^T A\mathbb{I}_1\Gamma_1 & \Gamma_4\mathbb{I}_4^T A\mathbb{I}_3\Gamma_3\mathbb{I}_3^T A\mathbb{I}_2\Gamma_2 + \Gamma_4\mathbb{I}_4^T A\mathbb{I}_2\Gamma_2 & \Gamma_4\mathbb{I}_4^T A\mathbb{I}_3\Gamma_3 & \Gamma_4
\end{bmatrix}$$

CA-mini-batch SDCA Step Sizes

$$
\begin{bmatrix} \mathbb{A}_1 \\ \mathbb{A}_2 \\ \mathbb{A}_3 \\ \mathbb{A}_4 \end{bmatrix} = \begin{bmatrix} \Gamma_1 & 0 & 0 & 0 \\ \Gamma_2 \mathbb{I}_2^T A \mathbb{I}_1 \Gamma_1 & \Gamma_2 & 0 & 0 \\ \Gamma_3 \mathbb{I}_3^T A \mathbb{I}_2 \Gamma_2 \mathbb{I}_2^T A \mathbb{I}_1 \Gamma_1 + \Gamma_3 \mathbb{I}_3^T A \mathbb{I}_1 \Gamma_1 & \Gamma_3 \mathbb{I}_3^T A \mathbb{I}_2 \Gamma_2 & \Gamma_3 & 0 \\ \Gamma_4 \mathbb{I}_4^T A \mathbb{I}_2 \Gamma_2 \mathbb{I}_2^T A \mathbb{I}_1 \Gamma_1 + \Gamma_4 \mathbb{I}_4^T A \mathbb{I}_3 (\Gamma_3 \mathbb{I}_3^T A \mathbb{I}_2 \Gamma_2 \mathbb{I}_2^T A \mathbb{I}_1 \Gamma_1 + \Gamma_3 \mathbb{I}_3^T A \mathbb{I}_1 \Gamma_1) + \Gamma_4 \mathbb{I}_4^T A \mathbb{I}_1 \Gamma_1 & \Gamma_4 \mathbb{I}_4^T A \mathbb{I}_3 \Gamma_3 \mathbb{I}_3^T A \mathbb{I}_2 \Gamma_2 + \Gamma_4 \mathbb{I}_4^T A \mathbb{I}_2 \Gamma_2 & \Gamma_4 \mathbb{I}_4^T A \mathbb{I}_3 \Gamma_3 & \Gamma_4 \end{bmatrix} \times \begin{bmatrix} \mathbb{I}_1^T r^{(0)} \\ \mathbb{I}_2^T r^{(0)} \\ \mathbb{I}_3^T r^{(0)} \\ \mathbb{I}_4^T r^{(0)} \end{bmatrix}
$$

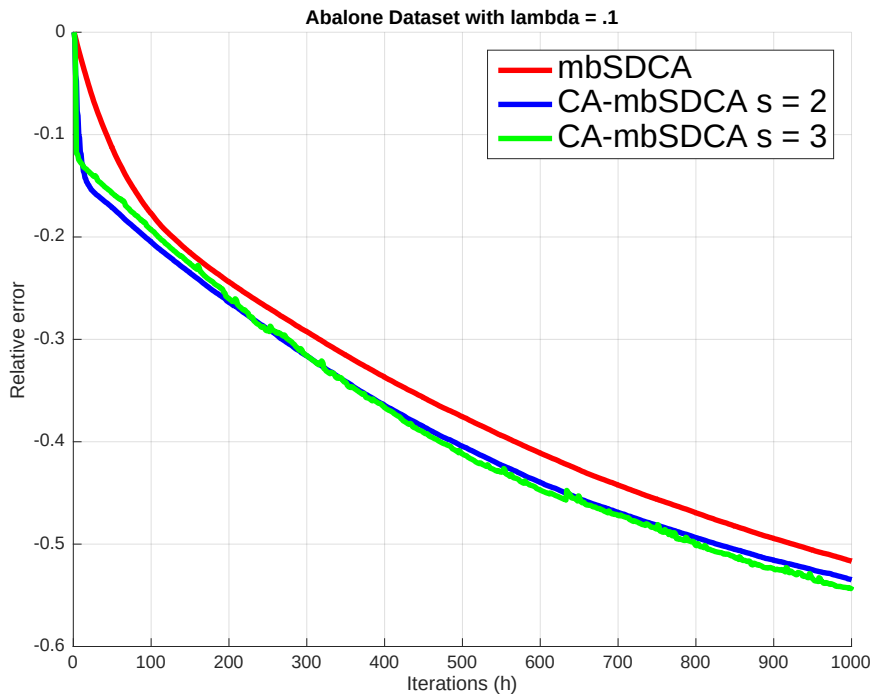
CA-mini-batch SDCA Communication Cost

The subroutine packs s iterations into Gram-matrix and matrix-vector computations.

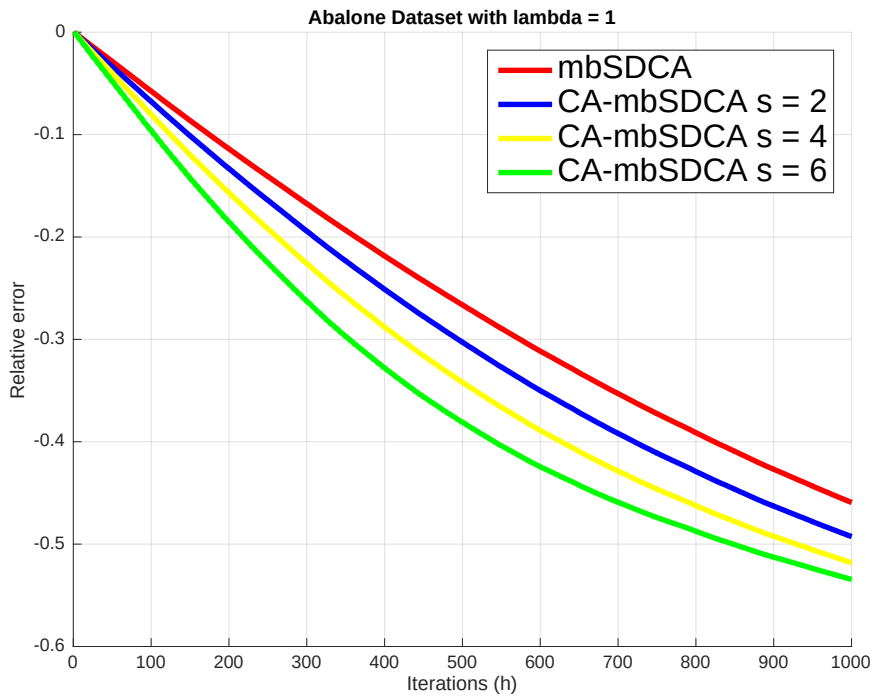
Algorithm	Latency (L)	Bandwidth (W)
mbSDCA	$O(H \log P)$	$O(Hb \log P + Hm \log P)$
CA-mbSDCA	$O(\frac{H}{s} \log P)$	$O(Hsm^2 \log P + \frac{Hm}{s} \log P)$

Reduces latency cost.
Reduces bandwidth cost if $(sb)^2 < m$.

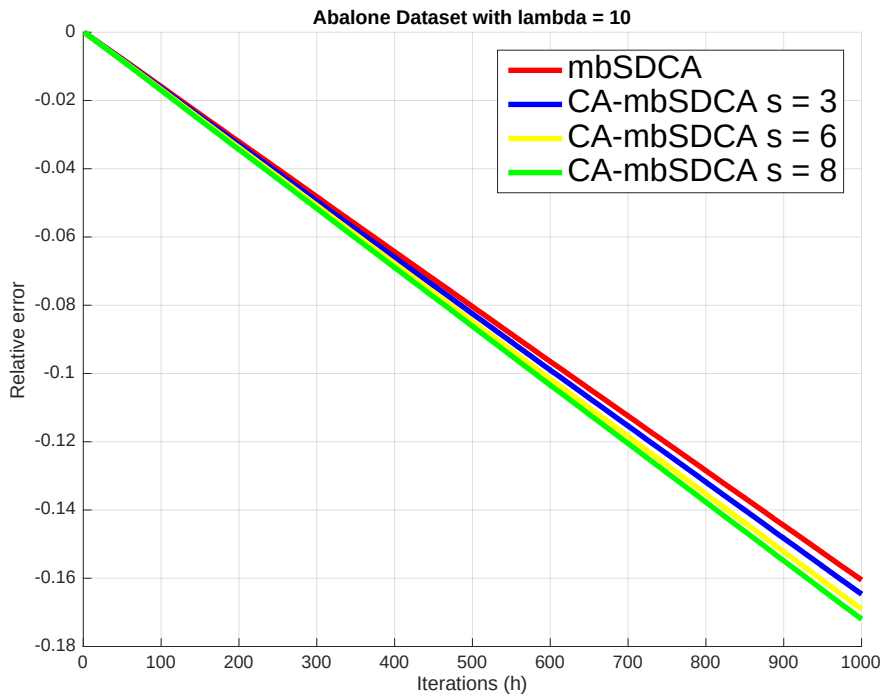
CA-mbSDCA Results: Abalone



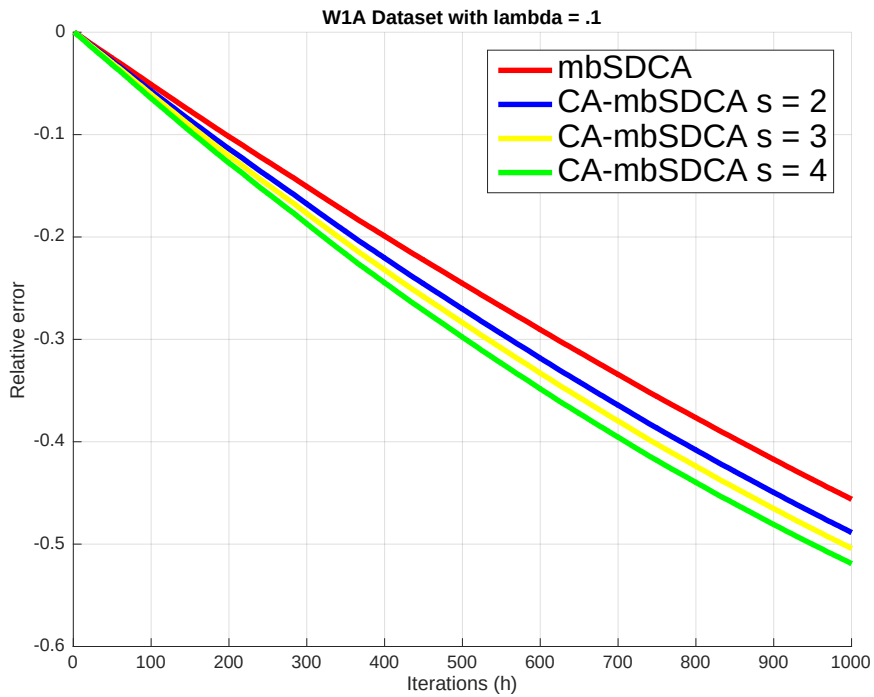
CA-mbSDCA Results: Abalone



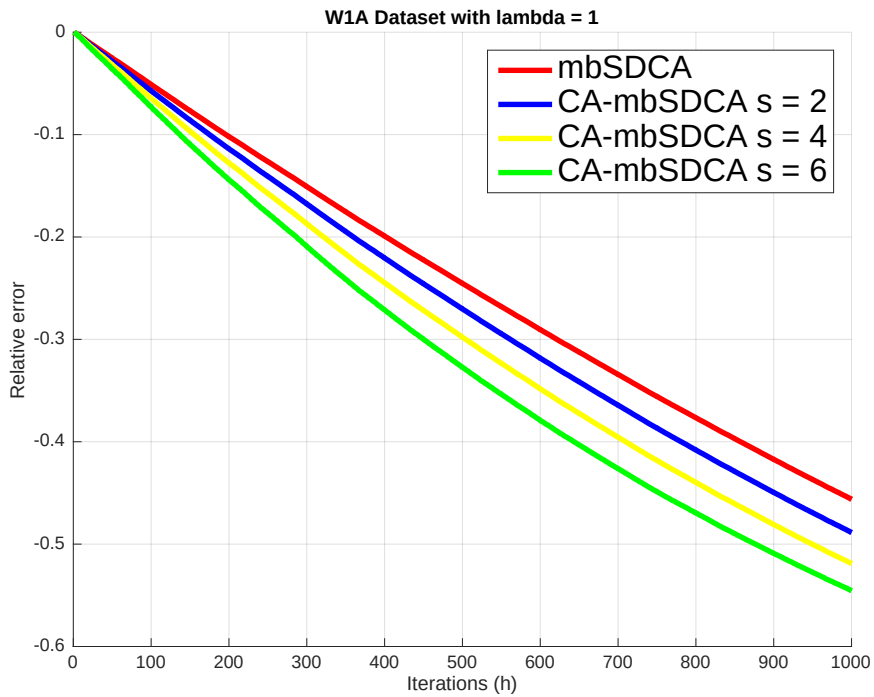
CA-mbSDCA Results: Abalone



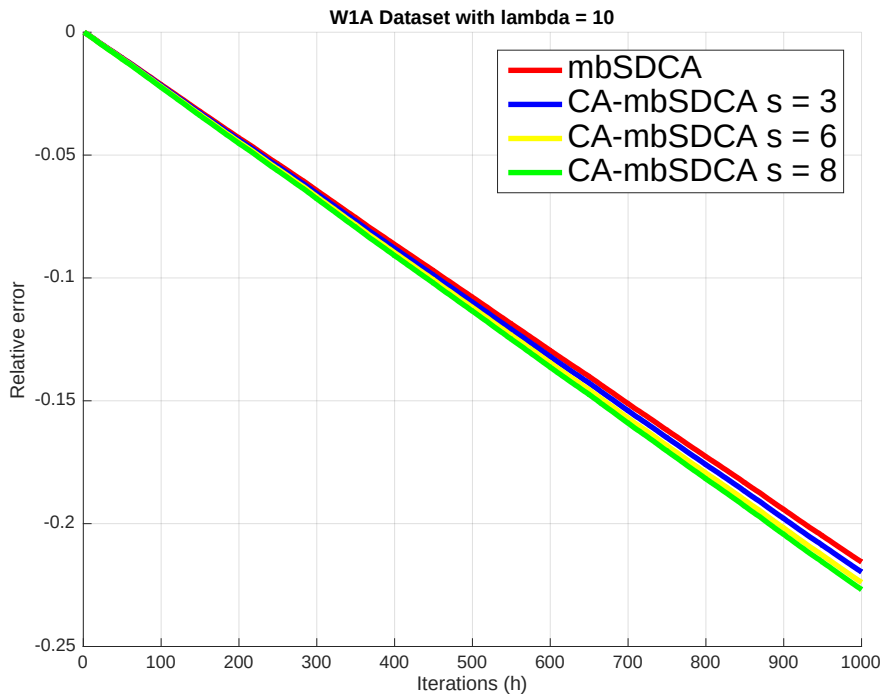
CA-mbSDCA Results: w1a



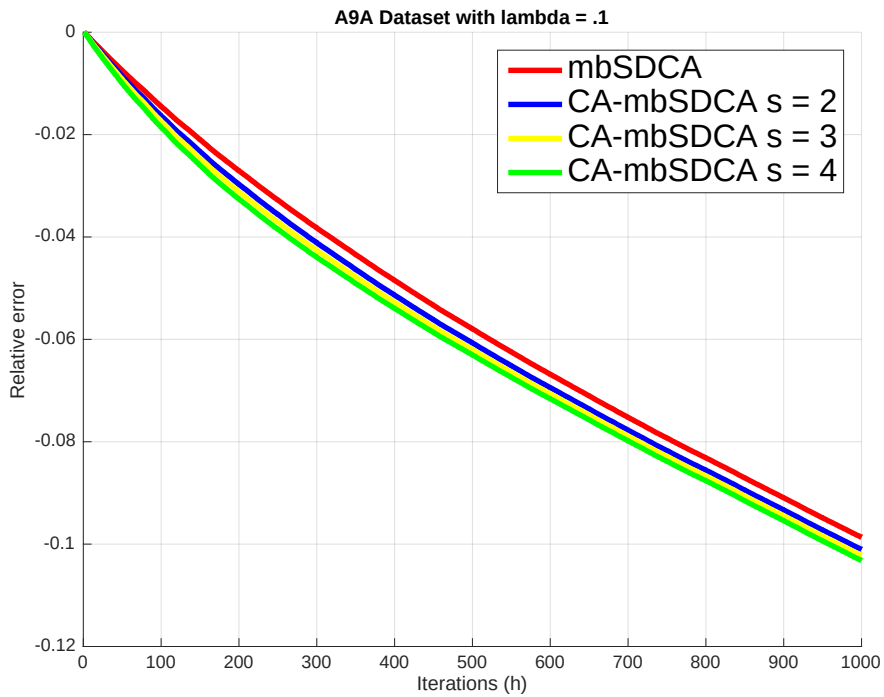
CA-mbSDCA Results: w1a



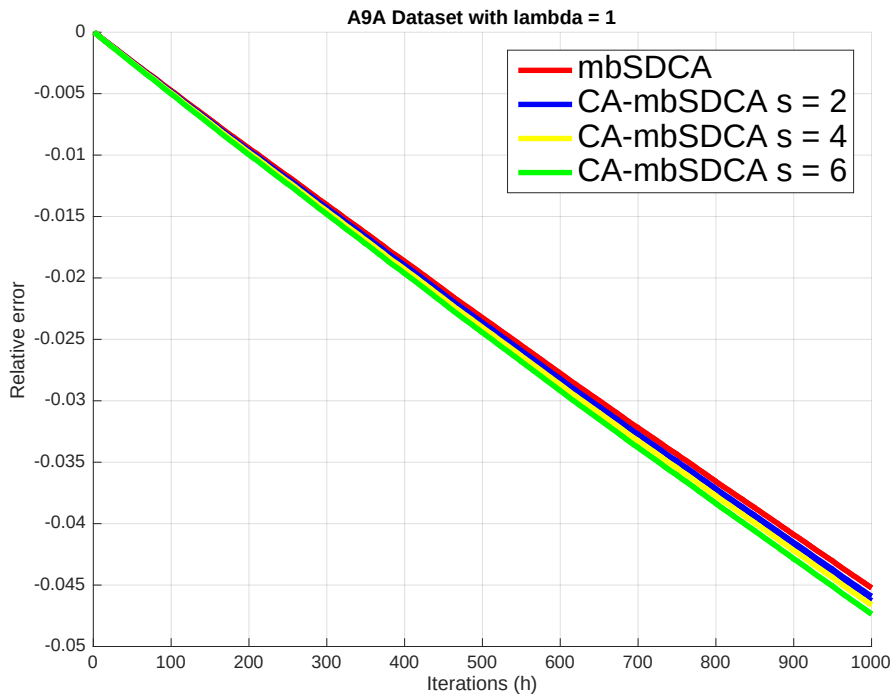
CA-mbSDCA Results: w1a



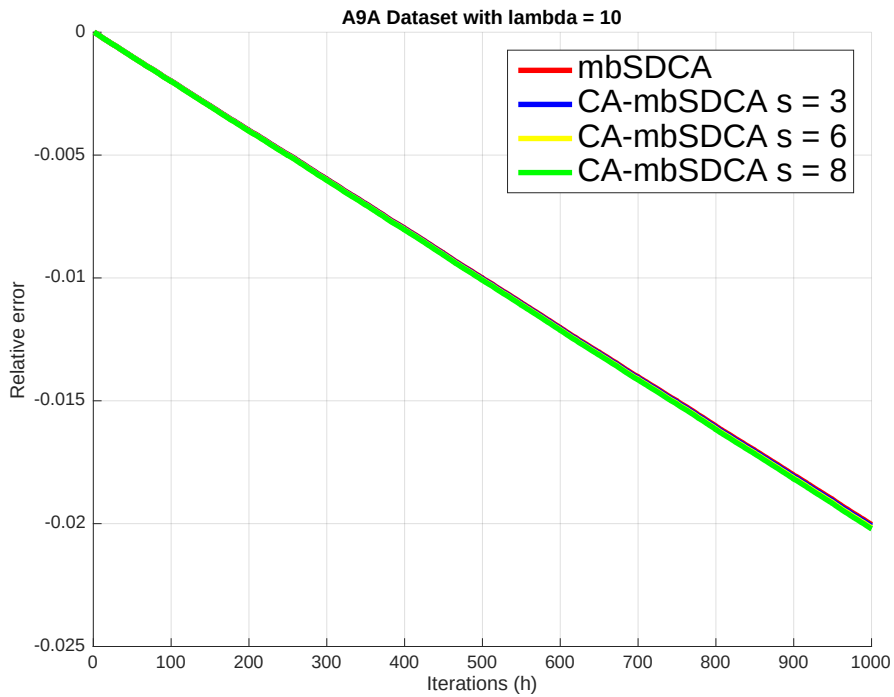
CA-mbSDCA Results: a9a



CA-mbSDCA Results: a9a



CA-mbSDCA Results: a9a



CA-SDCA Summary

Maximum attainable s depends on conditioning of X and λ.

CA-SDCA Summary

Maximum attainable s depends on conditioning of X and λ.

In practice, s and b should be chosen to balance:

- numerical error
- computation vs. communication

CA-SDCA Summary

Maximum attainable s depends on conditioning of X and λ.

In practice, s and b should be chosen to balance:

- numerical error
- computation vs. communication

Interesting to explore tradeoff between s and b.

Conclusion and Future Work

Derived CA versions of SDCA, mbSDCA and block-coordinate descent algorithms.

Experiments show these methods are numerically viable.

More work to be done!

Parallel implementations in progress.

Need to extend to L1-regularization and non-linear loss functions.