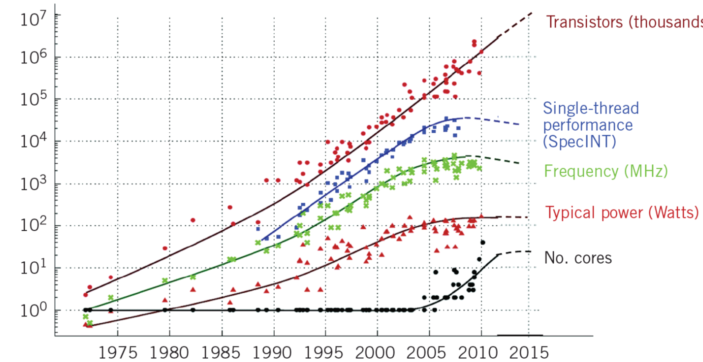
Sparsity and asynchrony in distributed optimization: models and convergence results

Arda Aytekin, Hamid Reza Feyzmahdavian, Sarit Khirirat and Mikael Johansson
KTH - Royal Institute of Technology

Achiving scalability in a post-Moore era

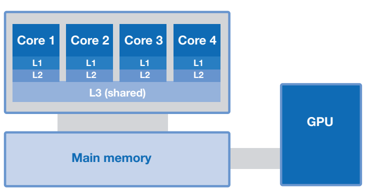
Single-thread performance increases are long gone

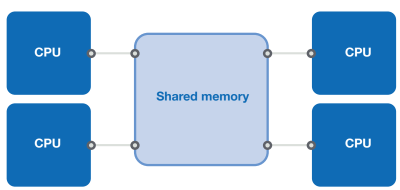


Key is now **more processing elements** (threads, cores, sockets, . . .)

Multi-core computing

Multiple computation units (cores) able to address the same memory space





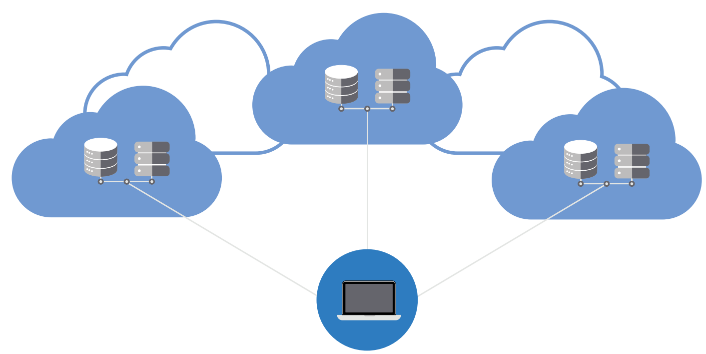
Many uses in optimization

- parallelize linear algebra, evaluate gradients in parallel, . . .

Critical to keep cores busy, respect memory hierarchies & bus limitations

Dealing with the data deluge

Increasingly often impossible/impractical to move data to central location

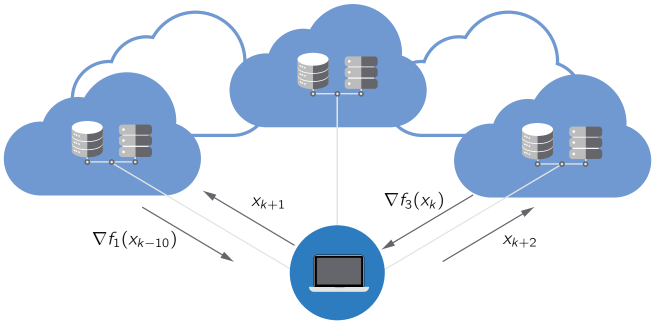


Geographically dispersed data, **heterogenous** compute resources

Dealing with the data deluge

Natural with master-worker solutions:

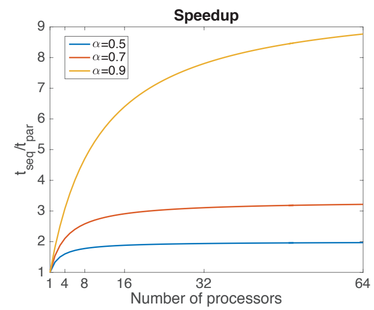
- master maintains decision vector, queries workers in parallel
- workers return **delayed** gradients of their data loss

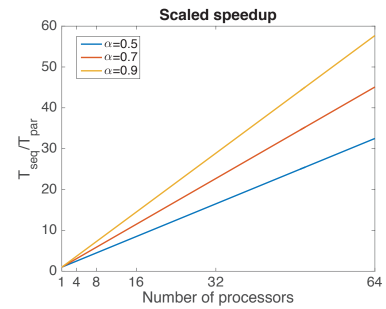


Q: What is the impact of time-varying delays on the algorithm convergence?

Limits of scalability

Speed-ups limited by fraction of code α which is parallelizable.





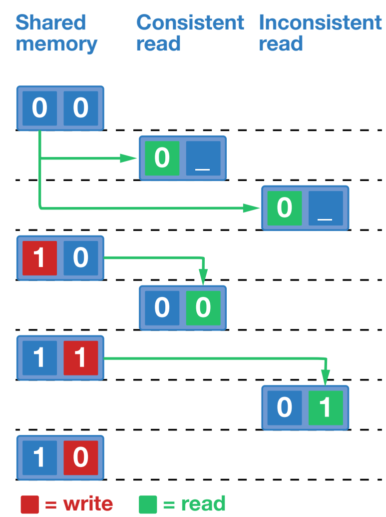
Idealized behaviors, further impaired by

- synchronization and lock management, communication, load imbalance

(challenges on multi-cores and clouds are surprisingly similar)

Contents

Lock-free implementations: consistent and inconsistent read



Time-delay models of asynchronism

Consistent read of vector x into variable z at time t:

- $z(t)$ has existed in shared memory at *some* time

$$z(t) = x(t - d(t))$$

homogeneous time delay for all components of z

Inconsistent read of x into z at time t:

- complete vector $z(t)$ has never existed in memory, only its components

$$z_i(t) = x_i(t - d_i(t))$$

heterogeneous delays

We will assume that information delays are bounded, arbitrarily time-varying.

Lyapunov analysis of synchronous algorithms

Convergence rates often derived using standard results for sequences.

Example. Gradient method with strongly convex objective satisfies

$$V_{k+1} \leq \rho V_k + r$$

which allows to conclude that $V_k \leq \rho^k V_0 + e$ where $e = r/(1-\rho)$.

Example. Gradient method for Lipschitz gradients analyzed by establishing

$$V_{k+1} \leq V_k - \alpha V_k^2$$

which implies that $V_k \leq V_0/(1 + \alpha k V_0)$.

Lyapunov analysis of asynchronous algorithms

Asynchronous algorithms result in sequences on the form

$$V_{k+1} \leq f(V_k, V_{k-d_k}) + e_k$$

Much harder to analyze, much less theoretical support.

Coming up: two sequence lemmas and an application

- allow for simple and uniform treatment of asynchronous algorithms
- balance simplicity, applicability and power; support analytical results

Convergence results for delayed sequences

Lemma 1. Let $\{V_k\}$ be a sequence of real numbers satisfying

$$V_{k+1} \leq pV_k + q \max_{k-d_k \leq j \leq k} V_j + r$$

for some non-negative numbers p, q and r. If $p + q < 1$ and

$$0 \leq d_k \leq d_{\max}$$

for all k, then

$$V_k \leq \rho^k V(0) + e$$

where $\rho = (p+q)^{1/(1+d_{\max})}$ and $e = r/(1-p-q)$.

[Feyzmahdavian, Aytekin and Johansson, 2014]

Convergence results for delayed sequences

Lemma 2. Assume that the non-negative sequences $\{V_k\}$ and $\{w_k\}$ satisfy

$$V_{k+1} \leq \rho V_k - b w_k + a \sum_{j=k-d_{\max}}^{k} w_j \,, \tag{1}$$

for some real numbers $\rho \in (0,1)$ and $a, b \geq 0$, and some integer $d_{\max} \geq 0$. Assume also that $w_k = 0$ for $k < 0$, and that

$$\frac{a}{1-\rho} \frac{1-\rho^{d_{\max}+1}}{\rho^{d_{\max}}} \leq b \,.$$

Then, $V_k \leq \rho^k V_0$ for all $k \geq 0$.

[Aytekin, Feyzmahdavian, Johansson, 2016]

Problem formulation

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \quad \sum_{i=1}^{m} f_i(x) + h(x)$$

- m samples, decision vector $x \in \mathbb{R}^n$
- $f_i(x)$ loss of sample i for decision x; $h(x)$ is regularizer

Assumptions:

- each f_i is convex, differentiable with Lipschitz continuous gradient
- $\sum_i f_i$ is strongly convex
- h is proper convex (but may be non-smooth, extended-real valued)

Examples: ℓ_1-regularized least-squares, constrained logistic regression, ...

The proximal incremental aggregate gradient algorithm

Idea:

- compute (incremental) gradient with respect to a subset of data
- maintain (aggregate of) most recent gradient for each data point
- update x using prox-step w.r.t aggregate gradient and regularizer

$$g_k = \sum_{i=1}^{m} \nabla f_i \left(x_{k-d_k^i} \right)$$

$$x_{k+1} = \underset{x}{\text{argmin}} \left\{ \langle g_k, x - x_k \rangle + \frac{1}{2\alpha} \|x - x_k\|_2^2 + h(x) \right\}.$$

Motivation: fewer calculations per iteration, faster wall-clock convergence (cf. SAG (Le Roux et al. 2012), IAG (Gürbüzbalaban et al. 2015), ...)

Related work

Blatt et al. (2007):

- convex quadratic loss, no regularizer, synchronous
- rate of convergence, but no explicit step-size or convegence factors

Tsen and Yun (2014)

- convex loss with Lipschitz gradient, simple regularizer, asynchronous
- rate of convergence, but no explicit step-size or convegence factors

Gürbüzbalaban et al. (2015)

- strongly convex loss with Lipschitz gradient, no regularizer, asynch.
- explicit step-sizes and convergence factors

and more (e.g. stochastic average gradient, ...)

Proximal incremental aggregate gradient on parameter server

$$g_k = \sum_{i=1}^{m} \nabla f_i \left(x_{k-d_k^i} \right) \tag{2}$$

$$x_{k+1} = \underset{x}{\text{argmin}} \left\{ \langle g_k, x - x_k \rangle + \frac{1}{2\alpha} \|x - x_k\|_2^2 + h(x) \right\}. \tag{3}$$

Natural parameter-server implementation:

- Data distributed over multiple workers $(\{1, \ldots, m\} = \mathcal{I}_1 \cup \mathcal{I}_2, \ldots)$
- Master node maintains iterate x, queries nodes for gradients

Time-varying, heterogeneous delays d_k^i between master and worker i.

Proximal incremental aggregate gradient on parameter server

Each worker w:

- receives new iterate from master, computes gradients of local data loss,

$$\sum_{i \in \mathcal{I}_w} \nabla f_i(x_k)$$

- pushes this quantity to master (arrives with total delay d_k^n)

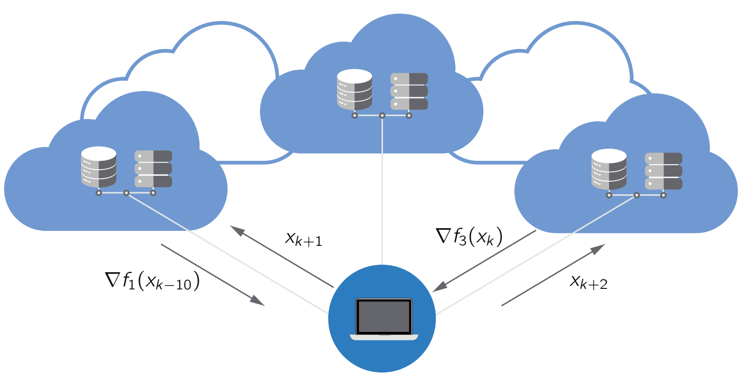
Master:

- maintains aggregate gradient

$$g_k = \sum_{i=1}^{m} \nabla f_i(x_{k-d_k^i})$$

- updates iterate via prox-step, pushes x_{k+1} to workers

PIAG on the parameter server



Main result

Theorem. Assume that each ∇f_i is L_i-Lipschitz continuous, $\sum_i f_i$ is μ-strongly convex, and $d_k^i \leq d_{\max}$ for all i. If the step-size α satisfies:

$$\alpha \leq \frac{\sqrt[d_{\max}+1]{1 + \frac{\mu}{L}\frac{1}{d_{\max}+1}} - 1}{\mu},$$

where $L = \sum_{n=1}^{N} L_n$, then the iterates generated by (2), (3) satisfy:

$$\|x_k - x^\star\|_2^2 \leq \left(\frac{1}{\mu\alpha + 1} \right)^k \|x_0 - x^\star\|_2^2 .$$

Discussion

Linear convergence, even in presence of proximal term.

In absence of asynchronism, can pick $\alpha = 1/L$ to guarantee

$$\|x_k - x^\star\|_2^2 \leq \left(\frac{L}{L+\mu}\right)^k \|x_0 - x^\star\|_2^2$$

Graceful slowdown guaranteed, as $d_{\max}$ increases

$$\rho \approx 1 - \frac{c}{(1+d_{\max})^2}$$

(similar to best known estimates for $h = 0$)

Sharper bounds, shorter and simpler proof than related work.

Proof sketch

Lemma 2. Assume that the non-negative sequences $\{V_k\}$ and $\{w_k\}$ satisfy

$$V_{k+1} \leq aV_k - bw_k + c \sum_{j=k-d_{\max}}^{k} w_j\,,$$

for some real numbers $a \in (0,1)$ and $b, c \geq 0$, and some integer $d_{\max} \geq 0$. Assume also that $w_k = 0$ for $k < 0$, and that the following holds:

$$\frac{c}{1-a}\,\frac{1-a^{d_{\max}+1}}{a^{d_{\max}}} \leq b\,.$$

Then, $V_k \leq a^k V_0$ for all $k \geq 0$.

Proof sketch

Convexity and Lipschitz continuity of gradients imply

$$\sum_{i=1}^{m} f_i(x_{k+1}) \leq \sum_{i=1}^{m} f_i(x) + \langle g_k, x_{k+1} - x\rangle + \sum_{i=1}^{m} \frac{L_i}{2}\|x_{k+1} - x_{k-d_k^i}\|_2^2 \quad \forall x$$

By strong convexity of $\sum_i f_i + h$, optimality conditions, and Jensen's ineq

$$\|x_{k+1} - x^\star\|_2^2 \leq \frac{1}{\mu\alpha+1}\|x_k - x^\star\|_2^2 - \frac{1}{\mu\alpha+1}\|x_{k+1} - x_k\|_2^2 + \\ + \frac{\alpha(d_{\max}+1)L}{\mu\alpha+1} \sum_{j=k-d_{\max}}^{k} \|x_{j+1} - x_j\|_2^2\,.$$

Now our Lemma applies and allows to conclude linear rate of convergence.

Parameter-server implementation on EC2

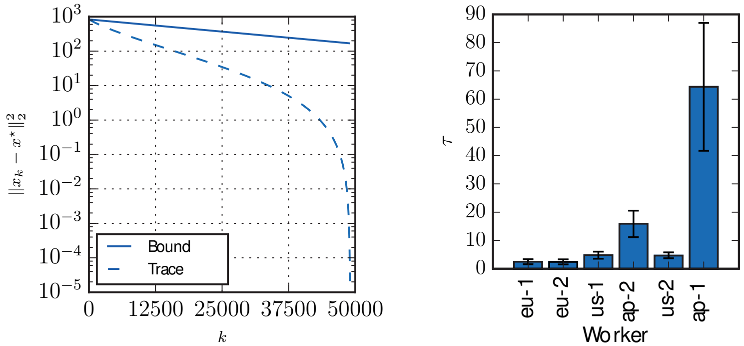
Binary classification via ℓ_1-regularized logistic regression on `rcv1-v2`

$$\underset{x}{\text{minimize}} \quad \frac{1}{m}\sum_{i=1}^{m}\left(\log\left(1+\exp\left(-b_i\langle a_i, x\rangle\right)\right) + \frac{\lambda_2}{2}\|x\|_2^2\right) + \lambda_1\|x\|_1\,,$$

Parameter-server implementation of (2), (3) on Amazon EC2:

- 3 compute nodes (`c4.2xlarge`: 8 CPUs, 15 GB RAM, each),
  - one in Ireland (EU),
  - one in North Virginia (US),
  - one in Tokyo (AP),
- 2 workers in each node (a total of 6 workers)
- Master node on computer at KTH in Stockholm, Sweden.

Parameter-server implementation on EC2



Parameter-server implementation on EC2

Amazon sent us the bill for the figure...

Computing:	\$ 80
Communication:	\$ 20

Computing far from free, communication surprisingly expensive.

Communication also impairs performance – important to reduce!

Contents

Data sparsity implies dimensionality reduction

Standard definition: many elements are zero (more than 66%)

- common feature of many large-scale data sets (e.g. in svmlib)

Standard implication: dimensionality reduction

- can store data more efficiently (row, col, val)
- approximate low-rank matrix representations

We will exploit another implication of sparsity...

Data sparsity implies decoupling

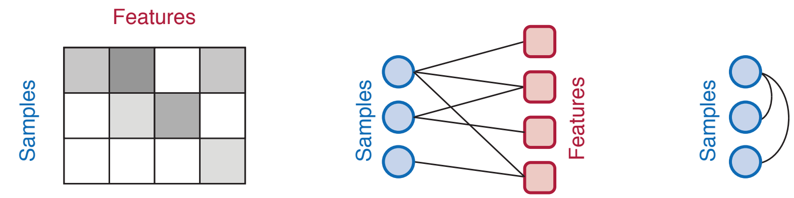
Example. Draw rows from matrix $A \in \mathbb{R}^{m \times n}$ with probability $1/m$.

$$\mathbb{E}\langle a_i, a_j \rangle \leq \mathbb{E}\|a_i\|_2^2$$

Inner product much smaller when A is sparse (can even be zero)!

How can we quantify and exploit this property?

Graphical representations of sparsity



Several graphical representations of sparsity

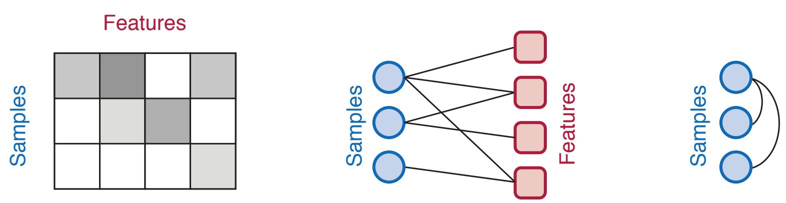
- bipartite sample-feature graph (edges if sample contains feature)
- sample conflict graph (edges if samples overlap in some feature)

(cf. Mania et al., arXiv:1507.06970)

Aim: use graphs to compute measure σ such that

$$\mathbf{E}\langle a_i, a_j \rangle \leq \sigma \mathbf{E}\|a_i\|_2^2$$

Graphical representations of sparsity



Key quantities:

- maximum feature degree $\Delta_r = \max_j |\{i : j \in \text{supp}(a_i)\}|$
- maximum or average conflict degree $\Delta_c^i = \sum_j \mathbf{1}\{\text{supp}(a_i) \cap \text{supp}(a_j) \neq 0$

With $\Delta_{\max} = \max_i \Delta_c^i$, and $\overline{\Delta}_c = \sum_i \Delta_c^i/m$, it holds that

$$\mathbf{E}\langle a_i, a_j \rangle \leq \min \left\{ \sqrt{\frac{1+\overline{\Delta}_c}{m}}, \frac{1+\Delta_{\max}}{m}, \sqrt{\frac{\Delta_r}{m}} \right\} \mathbf{E}\|a_i\|_2^2 := \sigma \mathbf{E}\|a_i\|_2^2$$

How sparse is real-world data?

Sparsity measure σ on data from libsvm (recall: $\mathbf{E}\langle a_i, a_j \rangle \leq \sigma \mathbf{E}\|a_i\|_2^2$)

Data set name	σ
kddb.t	0.255
w4a	0.61
rcv1	0.627
protein.t	0.669
news20	0.727

How can we use this sparsity in first-order methods?

Many machine-learning problem are on the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^m f_i(x) = \varphi(a_i^T x - b_i)$$

with $f_i(x) = \varphi(a_i^T x - b_i)$. Gradients have same sparsity pattern as data.

We will focus on mini-batch gradient descent:

$$x(t+1) = x(t) - \Gamma \sum_{i\in\mathcal{S}(t)} \gamma_i \nabla f_i(x)$$

where $\mathcal{S}(t)$ is a mini-batch of size M, drawn from $\{1,\dots,m\}$.

Mini-batch optimization under data sparsity

Assume that each f_i is L-Lipschitz continuous, total loss μ-strongly convex. Form mini-batch by sampling with replacement using probabilities $1/m$.

Mini-batch gradient descent generate iterates $\{x(t)\}$ which satisfy

$$\|x(t) - x^\star\|_2^2 \le \rho^t \|x(0) - x^\star\|_2^2 + e$$

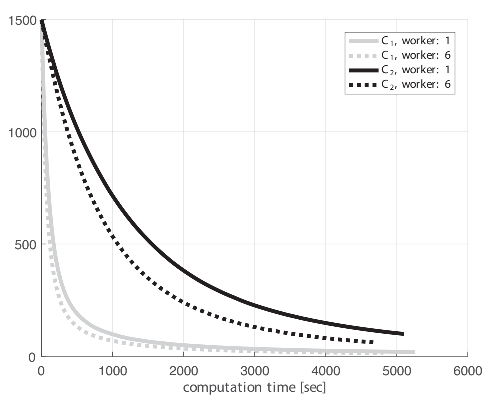
with

$$\rho = 1 - \frac{M}{1+(M-1)\sigma}\frac{\mu}{2mL}$$
$$e = \frac{1}{\mu L}\sum_i \|\nabla f_i(x^\star)\|_2^2$$

Recovers classical results in absence of sparsity, improvements when σ small.

Application to binary classification

Binary classification on data set with $m = 150000$, $n = 3000$ and $\Delta_r = 400$



Significant speed-ups by exploiting sparsity!

Many extensions

Can allow different Lipschitz constants, bias-convergence trade-off params.

Can derive similar results in absence of strong convexity.

Can deal with mini-batch proximal minimization for problems on the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^m f_i(x) + g(x)$$

Possible to combine with stochastic variance reduction (SVRG, etc.)

Pre-processing effort

Feature-degree practically for free.

Conflict graph very large, costly to form and manipulate

- some data set in libsvm takes about a day to analyze on standard PC
- tailored GPU code runs in more than 10x faster

Still, in practice, seems reasonable to focus on feature degree.

Conclusions

Scalability in a big-data, post-Moore world:

- parallel and distributed optimization
- exploiting structure, dealing with asynchronism, respecting architectures

Theory from lock-free and asynchronous computation

- two simple, yet powerful, sequence lemmas
- PIAG: convergence guarantees + cloud implementation

Exploiting data sparsity

- Graphical measures of data sparsity, evaluation on svmlib data
- Significant convergence guarantee improvements for mini-batch GD