Stochastic Asynchronous Parallel Methods in Optimization

Stephen Wright

University of Wisconsin-Madison

FoCM, December 2014

Collaborators: **Ji Liu** (U. Rochester), Victor Bittorf (Cloudera), Yijun Huang, Chris Ré (Stanford), Krishna Sridhar (GraphLab).

Formulations of Interest

$$\text{Unconstrained:} \quad \min_{x \in \mathbb{R}^n} f(x),$$

where f is at least continuous—also assume smooth, convex, strongly convex for some analysis.

$$\text{Summation Form:} \quad \min_{w \in \mathbb{R}^n} \phi(w) := \frac{1}{m} \sum_{i=1}^{m} \phi_i(w),$$

where m may be very large.

$$\text{Structured Regularization:} \quad \min_{x \in \mathbb{R}^n} f(x) + \lambda \Omega(x),$$

where Ω is convex, nonsmooth, but typically separable or block-separable:

$$\Omega(x) = \sum_{i=1}^{n} \Omega_i(x_i). \qquad \text{(example: } \Omega(x) = \|x\|_1\text{)}$$

Basic Coordinate Descent (CD) Framework

... for smooth unconstrained minimization: $\min_x f(x)$:

```
Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
**repeat**
    Choose index $i_k \in \{1, 2, \dots, n\}$;
    $x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$;
    $k \leftarrow k + 1$;
**until** termination test satisfied;
```

where

- $e_i = (0, \dots, 0, 1, 0, \dots, 0)^T$: the ith coordinate vector;
- $[\nabla f(x)]_i = i$th component of the gradient $\nabla f(x)$;
- $\alpha_k > 0$ is the step length.

Many variants within this framework, including block CD, where each i_k is a subset of $\{1, 2, \dots, n\}$ rather than a single index.

Basic Stochastic Gradient (SG) Framework

for summation form $\min_w \; \phi(w) := (1/m) \sum_{i=1}^m \phi_i(w)$:

```
Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
repeat
    Choose index $i_k \in \{1, 2, \dots, m\}$;
    $w^{k+1} \leftarrow w^k - \alpha_k \nabla \phi_{i_k}(w^k)$;
    $k \leftarrow k + 1$;
until termination test satisfied;
```

$\nabla \phi_{i_k}(w)$ is a proxy for $\nabla \phi(w)$.

Again, many variants.

The output of SG could be the latest iterate w^K, or a weighted average of all iterates $w^0, w^1, \dots, w^K$. The step between successive averaged iterates is a linear combination of all gradient estimates encountered so far:

$$\nabla \phi_{i_0}(w^0), \nabla \phi_{i_1}(w^1), \dots, \nabla \phi_{i_K}(w^{i_K}).$$

Applications: Summation Form

Extremely common form in data analysis and machine learning:

- support vector machines (primal form)
- least-squares, robust regression
- logistic regression (incl binary, multiclass, regularized)

In these applications ϕ_i often depends on a single item of data (e.g. a feature vector + observation), so $\nabla \phi_i$ may be much cheaper to evaluate than $\nabla \phi$, which requires the entire data set to be read.

The unknown vector w is a set of weights / coefficients.

Relating CD and SG

There is kind of duality relationship between CD and SG.

See this most evidently in the feasible linear system $Aw = b$, where A is $m \times n$ and possibly rank deficient: the Kaczmarz algorithm.

Write as a least-squares problem

$$\min_w \frac{1}{2m} \sum_{i=1}^{m} (A_i w - b_i)^2,$$

where A_i is the ith row of A (assume normalized: $\|A_i\|_2 = 1$ for all $i = 1, 2, \dots, m$).

SG updates with $\alpha_k \equiv 1$ are

$$w^{k+1} = w^k - A_{i_k}^T (A_{i_k} w^k - b_{i_k}) \quad \text{Kaczmarz step!}$$

Project onto the hyperplane define by the i_k equation.

Suppose we seek a minimum-norm solution from the formulation

$$\textbf{P:} \quad \min_{w \in \mathbb{R}^n} \frac{1}{2}\|w\|^2 \quad \text{subject to } Aw = b,$$

for which the dual is

$$\textbf{D:} \quad \min_{x \in \mathbb{R}^n} \frac{1}{2}\|A^T x\|_2^2 - b^T x,$$

where primal and dual solutions are related by $w = A^T x$.

CD updates applied to the dual with $\alpha_k \equiv 1$ are

$$x^{k+1} = x^k - (A_{i_k} A^T x^k - b_{i_k}) e_{i_k}.$$

Multiplying by A^T and using the identity $w = A^T x$, we obtain the Kaczmarz step again:

$$w^{k+1} = w^k - A_{i_k}^T (A_{i_k} w^k - b_{i_k}).$$

Randomized Kaczmarz: Convergence

Recall that $\|A_i\| = 1$ for all i. $\lambda_{\min,\text{nz}}$ denotes minimum nonzero eigenvalue of $A^T A$. $P(\cdot)$ is projection onto solution set.

$$\begin{aligned}
\frac{1}{2}\|x^{k+1} - P(x^{k+1})\|^2 &\le \frac{1}{2}\|x^k - A_{i_k}^T(A_{i_k}x^k - b_{i_k}) - P(x^k)\|^2 \\
&= \frac{1}{2}\|x^k - P(x^k)\|^2 - \frac{1}{2}(A_{i_k}x^k - b_{i_k})^2.
\end{aligned}$$

Suppose that i_k is chosen randomly with equal probability, and independently at each iteration. Take expectation over i_k:

$$\begin{aligned}
E\left[\frac{1}{2}\|x^{k+1} - P(x^{k+1})\|^2 \,|\, x^k\right] &\le \frac{1}{2}\|x^k - P(x^k)\|^2 - \frac{1}{2}E\left[(A_{i_k}x^k - b_{i_k})^2\right] \\
&= \frac{1}{2}\|x^k - P(x^k)\|^2 - \frac{1}{2m}\|Ax^k - b\|^2 \\
&\le \left(1 - \frac{\lambda_{\min,\text{nz}}}{m}\right)\frac{1}{2}\|x^k - P(x^k)\|^2.
\end{aligned}$$

[Strohmer and Vershynin, 2009]. Linear rate in expectation.

CD and SG for Empirical Risk Minimization (ERM)

The ERM framework [Lin et al., 2014] can express linear least-squares, support vector classification and regression, logistic regression etc:

$$\textbf{P:} \quad \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \phi_i(c_i^T w) + \lambda \Omega(w),$$

for vectors $c_i \in \mathbb{R}^d$ and convex functions ϕ_i and Ω, and regularization parameter λ. For conjugate function t^* defined as usual by

$$t^*(y) = \sup_z (z^T y - t(z)),$$

we can write the Fenchel dual of ERM as follows:

$$\textbf{D:} \quad \min_{x \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \phi_i^*(-x_i) + \lambda \Omega^* \left(\frac{1}{\lambda n} C x \right),$$

where C is the $d \times n$ matrix whose columns are c_i.

SG on **P** corresponds to CD on **D** e.g. for $\Omega(s) = (1/2) s^T s$.

Coordinate Descent

Consider the simplest setting: single-coordinate descent for

$$\min_{x \in \mathbb{R}^n} f(x)$$

where f is a smooth continuous function.

Often OK for applications — and necessary for analysis — to make additional assumptions on f, such as

- smooth, e.g. Lipschitz continuously differentiable
- convex
- strongly convex

(Discuss extensions to structured nonsmooth objectives and block-coordinate updates later.)

Variants

Selection of coordinate i_k:

- Randomized: Select i_k at random and independently at each iteration.
- Cycle through the coordinates: $i_{k+1} = [(i_k + 1) \mod n] + 1$;
- "Essentially cyclic:" touch each coordinate i at least once in each stretch of T iterations;

Line Search α_k:

- Short step: α_k prescribed, using global knowledge about f (bounds on Lipschitz constant for ∇f, modulus of convexity);
- Line search: Choose α_k to satisfy "sufficient decrease" conditions for f along the i_k coordinate direction;
- Exact: Choose α_k to exactly minimize f along i_k coordinate.

Coordinate Descent (CD): Background

CD methods have been popular with practitioners over many years:

- Intuitive: Solve an optimization problem by solving a sequence of easier problems — in this case, one-dimensional optimization.
- Often easy to implement.
- Handle bounds and regularization well.

There was little interest among optimization researchers until recently (with some very notable exceptions: Bertsekas, Tseng, Luo,….)

- Convergence analysis of popular variants was not obvious!
- According to certain assumptions on the cost of evaluating derivatives, it's not clear if CD is competitive in theory.

But interest among optimization specialists has grown since 2009, and new applications appear steadily.

Applications: Coordinate Descent

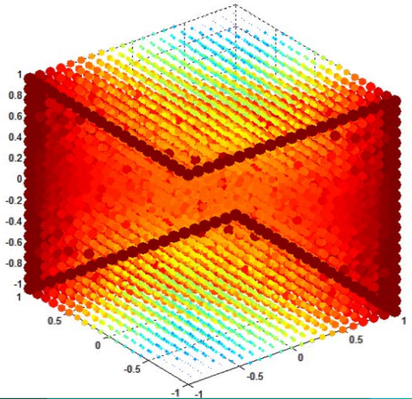
From the literature 1990-2014:

- Support vector machines (dual formulation, with kernel).
- Positron emission tomography, optical diffusion tomography.
- Protein structure - adjusting dihedral angles in a protein chain so that the end of the chain is in a specified position.
- Gene expression studies (via logistic regression).
- Recovering origin-destination matrices from traffic observations.
- Functional MRI image analysis.
- Generalized linear models in statistics.
- Transceiver design via tensor optimization.
- Phase retrieval in X-ray crystallography.
- Self-calibrating sensing models: $y = A(\theta)x$.

Convergence? Not Always: [Powell, 1973]

$$f(x_1, x_2, x_3) = -(x_1x_2 + x_2x_3 + x_1x_3) + \sum_{i=1}^{3}(|x_i| - 1)_+^2.$$

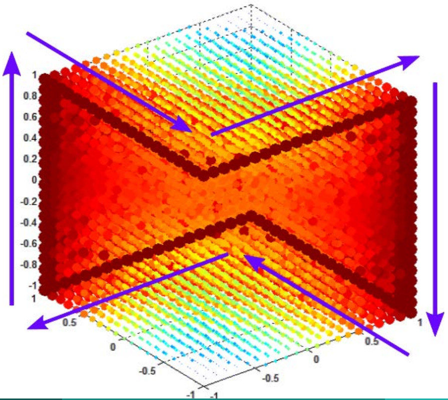
Nonconvex. Minimizers at $(1, 1, 1)^T$ and $(-1, -1, -1)^T$. CD with cyclic ordering, exact minimization cycles near the six nonoptimal vertices.



Convergence? Not Always: [Powell, 1973]

$$f(x_1, x_2, x_3) = -(x_1x_2 + x_2x_3 + x_1x_3) + \sum_{i=1}^{3}(|x_i| - 1)_+^2.$$

Nonconvex. Minimizers at $(1,1,1)^T$ and $(-1,-1,-1)^T$. CD with cyclic ordering, exact minimization cycles near the six nonoptimal vertices.



Convergence: Convex, Randomized, Short-Step

Convergence can be proved for the case of smooth, convex f.

[Nesterov, 2009] showed that randomized, short-step methods can be analyzed in a similar fashion to the corresponding full-gradient methods.

```
SCD (Stochastic Coordinate Descent)
Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
repeat
    Choose index $i_k \in \{1, 2, \ldots, n\}$ at random with equal probability;
    Set $\alpha_k \equiv 1/L_{\max}$;
    $x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$;
    $k \leftarrow k + 1$;
until termination test satisfied;
```

Here $L_{\max}$ is a componentwise Lipschitz constant for ∇f:

$$|[\nabla f(x + te_i)]_i - [\nabla f(x)]_i| \leq L_{\max}|t|, \quad \text{for all } x,\ t,\ i.$$

Convergence: Full-Gradient vs Coordinate Descent

Classical short-step full-gradient descent method for minimizing f:

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k), \quad \text{where } \alpha_k \equiv 1/L,$$

where L is the Lipschitz constant for ∇f, that is,

$$\|\nabla f(x+d) - \nabla f(x)\| \le L\|d\|, \quad \text{for all } x, \ d.$$

All iterates x^k lie in the level set $S(x^0) := \{x \,|\, f(x) \le f(x^0)\}$. Suppose there is R_0 such that $\|x - x^*\| \le R_0$ for all $x \in S(x^0)$. Then can show

$$f(x^k) - f^* \le \frac{2LR_0^2}{k}.$$

Sublinear convergence with rate $1/k$.

Similar analysis for SCD yields an expected sublinear rate

$$E[f(x^k) - f^*] \le \frac{2nL_{\max}R_0^2}{k}.$$

(Expectation taken over the random coordinates: $i_0, i_1, i_2, \dots$.)

Full-Gradient vs Coordinate Descent

The SCD convergence rate from the full-gradient rate in that

- extra factor of n — not surprising, as we are using only "one-nth" of the full gradient;
- $L_{\max}$ replaces L (possibly $L_{\max} < L$).

[Beck and Tetruashvili, 2013] derive a similar rate for Cyclic CD: With steps $\alpha_k \equiv L$ obtain ($L =$ usual Lipschitz constant) have

$$f(x^k) - f^* \leq \frac{4Ln(n+1)R_0^2}{k+8}, \quad \text{for } k = n, 2n, 3n, \dots .$$

(The dependence on n is slightly worse, but it may or may not have a bearing on practical performance.)

Convergence: Strongly Convex f

Suppose f is strongly convex with modulus σ:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\sigma}{2} \|y - x\|^2$$

SCD with steps $\alpha_k \equiv 1/L_{\max}$ yields convergence at a linear rate:

$$E[f(x^k) - f^*] \leq \left(1 - \frac{\sigma}{nL_{\max}}\right)^k [f(x^0) - f^*].$$

Again, similar to the full-gradient rate in this case, except for the factor of n and the different Lipschitz constant.

Cyclic CD with $\alpha_k = 1/L$ also yields a similar linear rate [Beck and Tetruashvili, 2013]:

$$f(x^k) - f^* \leq \left(1 - \frac{\sigma}{2(n+1)L}\right)^{k/n} [f(x^0) - f^*], \quad k = n, 2n, 3n, \dots .$$

Similar except for different Lipschitz constant and extra factor of n in the rate (due to the $1/n$ power).

Sublinear vs Linear Convergence

Sublinear convergence (e.g. error decreases like C/k or C/k^2 in iteration number k, for some constant C) is slow.

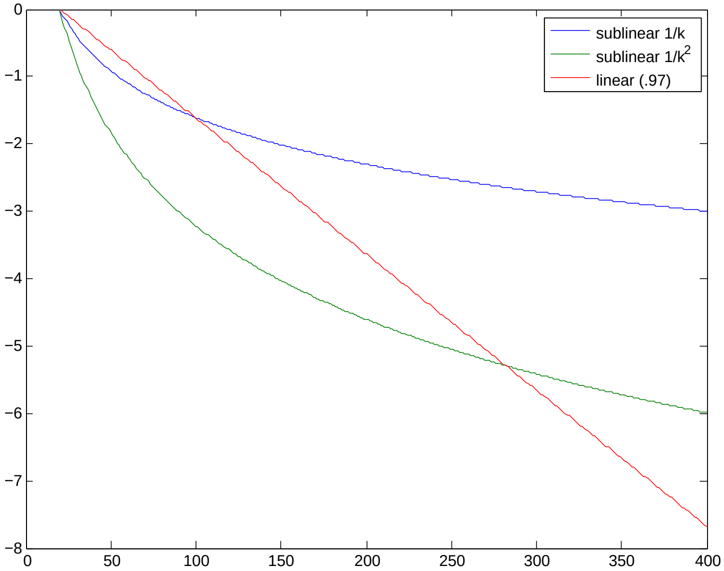
To achieve a specified accuracy ϵ, need $O(C/\epsilon)$ iterations for a $1/k$ process, and $O(\sqrt{C/\epsilon})$ for a $1/k^2$ process.

Linear convergence is asymptotically faster. If error decreases like $(1-\delta)^k$ for some small positive δ, need $O(|\log \epsilon|/\delta)$ iterates to reach accuracy ϵ.

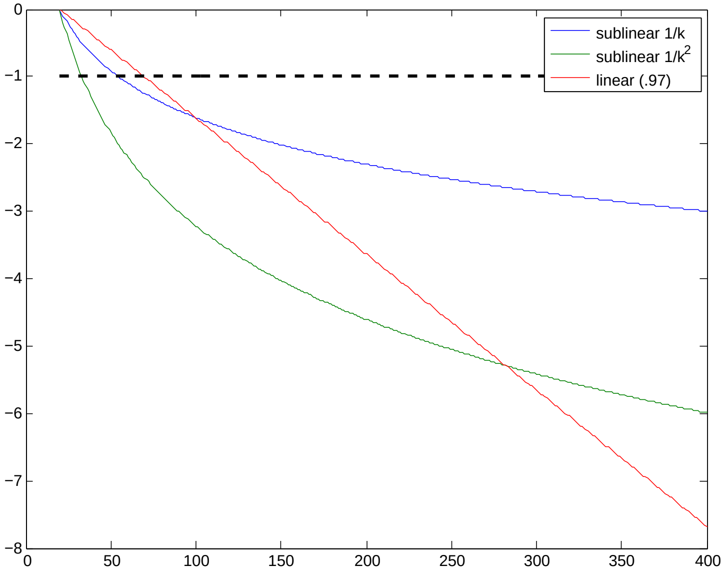
Sublinear rates were often viewed almost as uninteresting. But some modern applications need only a low-accuracy solution, and a sublinear algorithm may suffice.

(The constants in the $O()$ terms and the cost per iteration are significant.)

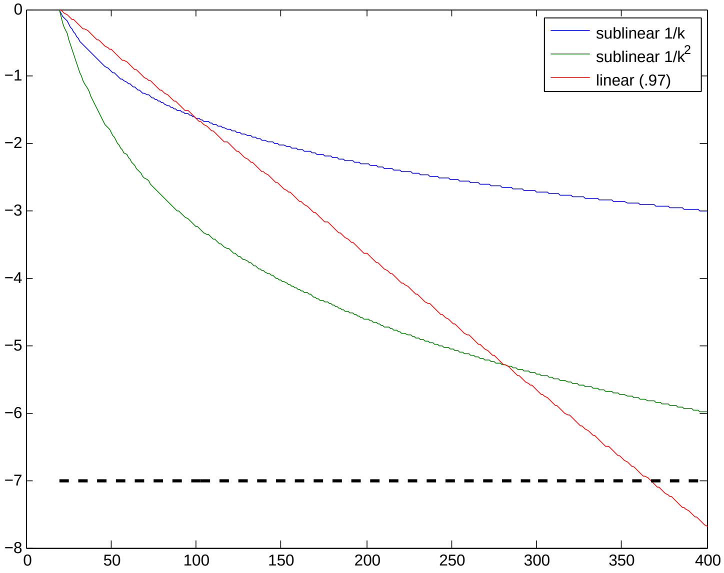
Sublinear vs Linear Convergence: Semilog Plot



Sublinear vs Linear Convergence: Semilog Plot



Sublinear vs Linear Convergence: Semilog Plot



Asynchronous Parallel Optimization

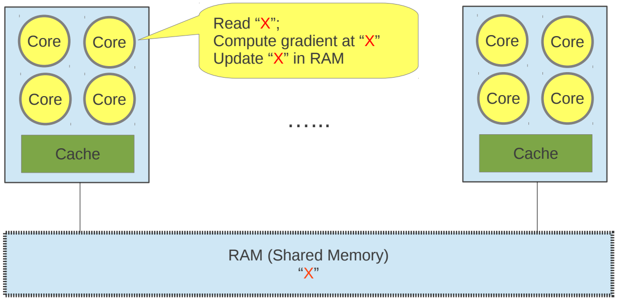


Figure: *Asynchronous* parallel setup used in HOGWILD! [Niu et al., 2011]

- All cores share the same memory, containing the variable x;
- All cores run the same optimization process independently;
- All cores read and update the coordinates of x concurrently without any hardware or software locking — no coordination!

Use the same model of computation for the algorithms discussed below.

Asynchronous Random Kaczmarz [Liu et al., 2014]

Each core runs the following process, for given stepsize $\gamma > 0$:

```
repeat
    • Choose index $i \in \{1, 2, \ldots, n\}$ uniformly at random;
    • Choose component $t \in \text{supp}(A_i)$ uniformly at random;
    • Read the supp($A_i$)-components of $x$ (from shared memory);
    • Update the $t$ component of $x$:
          $x_t \leftarrow x_t - \gamma \|A_i\|_0 (A_i)_t (A_i x - b_i)$
      (unitary operation);
until whenever;
```

Note that x can be updated by other cores between the time it is read and the time that the update is performed.

Randomized Kaczmarz analysis does not apply directly, as each update is using outdated information and we update just a single component of x.

AsyRK: Global View

From a central, global viewpoint, aggregating the actions of the individual cores, we have the following:

```
• Choose $x^0$ and steplength $\gamma > 0$;
for $k = 0, 1, 2, \dots$ do
    • Choose index $i_k \in \{1, 2, \dots, n\}$ uniformly at random;
    • Choose component $t_k \in \text{supp}(A_{i_k})$ uniformly at random;
    • Update the $t_k$ component of $x^k$:
        $x^{k+1} \leftarrow x^k - \gamma \|A_{i_k}\|_0 (A_{i_k})_{t_k} (A_{i_k} x^{j(k)} - b_{i_k})$
      (unitary operation);
end for
```

where $j(k)$ is some iterate prior to k but no more than τ cycles old:

$$k - j(k) \leq \tau.$$

If all computational cores are roughly the same speed, we can think of the delay τ as being similar to the number of cores.

Consistent Reading

Assumes consistent reading, that is, the $x^{j(k)}$ used in the step calculation is an "x" that actually existed at some point in the shared memory.

(This condition may be violated if two or more updates happen to the supp(A_{i_k})-components of x while they are being read.)

When the vectors A_i are sparse, inconsistency is not too frequent.

More on this later!

AsyRK Analysis: A Key Element

Key parameters:

- $\mu := \max_{i=1,2,\dots,m} \|A_i\|_0$ (maximum nonzero row count);
- $\alpha := \max_{i,t} \|A_i\|_0 \|(Ae_t)A_{i,t}\| \le \mu \|A\|$;
- $\lambda_{\max} =$ max eigenvalue of $A^T A$.

Idea of analysis: Choose some $\rho > 1$ and choose steplength γ small enough that

$$\rho^{-1}\mathbb{E}(\|Ax^k - b\|^2) \le \mathbb{E}(\|Ax^{k+1} - b\|^2) \le \rho\mathbb{E}(\|Ax^k - b\|^2).$$

Not too much change to the residual at each iteration. Hence, don't pay too much of a price for using outdated information.

But don't want γ to be too tiny, otherwise overall progress is too slow.

Strike a balance!

Main Theorem

Theorem

Choose any $\rho > 1$ and define γ via the following:

$$\psi = \mu + \frac{2\lambda_{\max}\tau\rho^{\tau}}{m}$$

$$\gamma \le \min\left\{\frac{1}{\psi},\ \frac{m(\rho-1)}{2\lambda_{\max}\rho^{\tau+1}},\ m\sqrt{\frac{\rho-1}{\rho^{\tau}(m\alpha^2+\lambda_{\max}^2\tau\rho^{\tau})}}\right\}$$

Then have

$$\rho^{-1}\mathbb{E}(\|Ax^k-b\|^2) \le \mathbb{E}(\|Ax^{k+1}-b\|^2) \le \rho\mathbb{E}(\|Ax^k-b\|^2)$$

$$\mathbb{E}(\|x^{k+1}-P(x^{k+1})\|^2) \le \left(1-\frac{\lambda_{\mathrm{min,nz}}\gamma}{m\mu}(2-\gamma\psi)\right)\mathbb{E}(\|x^k-P(x^k)\|^2),$$

A particular choice of ρ leads to simplified results, in a reasonable regime.

A Particular Choice

Corollary

Assume

$$2e\lambda_{\max}(\tau+1) \le m$$

and set $\rho = 1 + 2e\lambda_{\max}/m$. Can show that $\gamma = 1/\psi$ for this case, so expected convergence is

$$\mathbb{E}(\|x^{k+1} - P(x^{k+1})\|^2) \le \left(1 - \frac{\lambda_{\mathrm{min,nz}}}{m(\mu+1)}\right) \mathbb{E}(\|x^k - P(x^k)\|^2).$$

In the regime $2e\lambda_{\max}(\tau+1) \le m$ considered here the delay τ doesn't really interfere with convergence rate. In this regime, speedup is linear in the number of cores!

Rate is consistent with serial randomized Kaczmarz

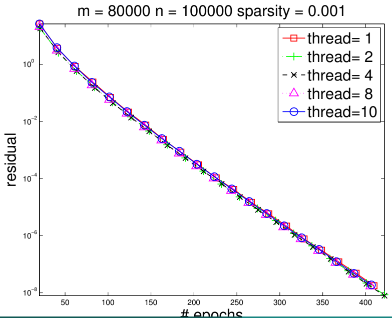
AsyRK: Near-Linear Speedup

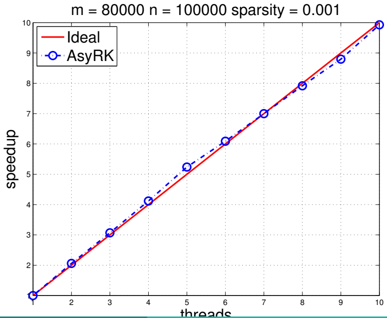
Run on an Intel Xeon 40-core machine. Used one socket — 10 cores.

Diverges a bit from the analysis:

- We update *all* components of x for $A_{i_k}^T$ (not just component t);
- Choose i_k randomly, but use sampling without replacement to work through the rows of A, reordering after each "epoch"

Sparse Gaussian random matrix $A \in \mathbb{R}^{m \times n}$ with $m = 100000$ and $n = 80000$, sparsity $\delta = .001$. See linear speedup.

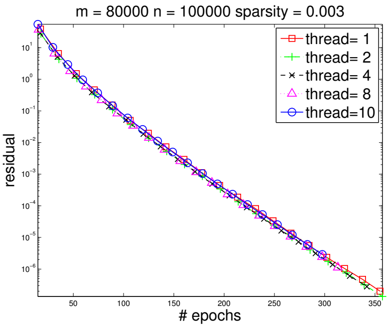


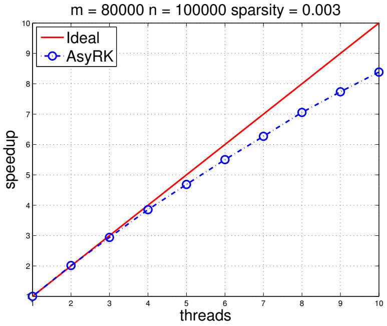


AsyRK: Near-Linear Speedup

Sparse Gaussian random matrix $A \in \mathbb{R}^{m \times n}$ with $m = 100000$ and $n = 80000$, sparsity $\delta = .003$.

See slight dropoff from linear speedup for this slightly less-sparse problem.

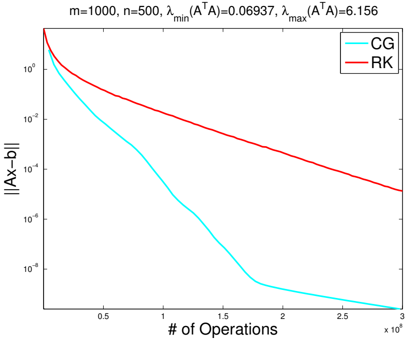


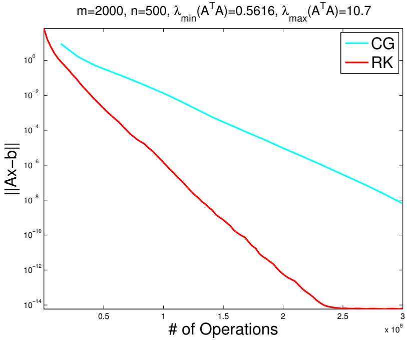


(Runtime: 18.4 seconds on 10 cores.)

RK vs Conjugate Gradient

Compare serial implementations of RK and CG (applied to $A^T A$). Multicore speedups are similar for both. Random A, sparsity $\delta = .1$.





Comparison of asymptotic linear rates shows that RK becomes competitive for $\lambda_{\text{max}}\lambda_{\min} > 1$, where these are the max / min eigenvalues of $A^T A$. (Recall: rows of A are normalized).

Regularized Objectives

Many modern applications of optimization seek *approximate* minimizers of f with *desirable structure*. One way to achieve this is to add a *regularizer* $\Omega(x)$ to the objective:

$$\min_x f(x) + \lambda \Omega(x).$$

- $\Omega(x)$ is chosen to impose the desired structure on x;
- $\lambda \geq 0$ is a *regularization parameter* that controls the amount of regularization.

$$\lambda \text{ large} \Rightarrow \text{more emphasis on structure;}$$
$$\lambda \text{ small} \Rightarrow \text{more emphasis on optimizing } f.$$

Regularized Formulation

$$\min_x : \ F(x) := f(x) + \lambda\Omega(x) \tag{1}$$

- $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ is convex and differentiable;
- $\Omega(\cdot) : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ is a proper closed convex real value extended function;
- $\Omega(x)$ is separable: $\Omega(x) = \sum_{i=1}^n \Omega_i(x_i)$, $\Omega_i(\cdot) : \mathbb{R} \mapsto \mathbb{R} \cup \{+\infty\}$.

Instances of $\Omega(x)$:

- Box-constrained: $\Omega(x) = \sum_{i=1}^n \mathbf{1}_{[a_i,b_i]}(x_i)$ where $\mathbf{1}_{[a_i,b_i]}$ is an indicator function for $[a_i, b_i]$;
- ℓ_p norm regularization for $p \geq 1$: $\Omega(x) = \|x\|_p^p$.
  (The ℓ_1 norm $\|x\|_1$ is especially popular.)

Stochastic Proximal Coordinate Descent SPCD

Define prox-operator $\mathcal{P}_h$ for a convex function h:

$$\mathcal{P}_h(y) = \arg\min_x \frac{1}{2}\|x-y\|^2 + h(x).$$

(It's nonexpansive: $\|\mathcal{P}_h(y) - \mathcal{P}_h(z)\| \le \|y-z\|$.)

Basic Step: Select a coordinate i and compute the gradient element $[\nabla f(x)]_i$; take a step along this direction and "shrink" to account for coordinate regularizer Ω_i.

$$x_i \leftarrow \mathcal{P}_{\alpha\lambda\Omega_i}(\underbrace{x_i - \alpha[\nabla f(x)]_i}_{\text{coordinate gradient}}),$$

for some step length α.

This is equivalent to solving an approximate version of the coordinate-i problem in which f is replaced by a simple quadratic:

$$\min_{z_i} \nabla_i f(x)^T [z_i - x_i] + \frac{1}{2\alpha}[z_i - x_i]^2 + \lambda\Omega_i(z_i).$$

Prox-Operator Examples

Prox-operators can be executed efficiently in many cases.

- $\Omega_i(t) = |t|$: soft thresholding operation

$$\mathcal{P}_{\lambda\Omega_i}(t) = \operatorname{sgn}(t)\max\{|t| - \lambda, 0\}.$$

- $\Omega_i(t) = \mathbf{1}_{[a,b]}$: projection operation

$$\mathcal{P}_{\lambda\Omega_i}(t) = \arg\min_{s\in[a,b]} \frac{1}{2}\|s - t\|^2 = \operatorname{mid}(a, b, t).$$

- Ω_i null (no regularization on component i): Then $\mathcal{P}_{\lambda\Omega_i}(t) = t$. The basic CD step is then

$$x_i \leftarrow x_i - \alpha[\nabla f(x)]_i,$$

so reduces to the basic CD step for a smooth function.

Asynchronous Parallel Stochastic Proximal Coordinate Descent Algorithm (AsySPCD)

All processors run a stochastic proximal coordinate descent process concurrently and without synchronization, for some $\gamma > 0$:

```
**repeat**
    • Select a coordinate $i \in \{1, 2, \dots, n\}$ uniformly at random;
    • Read "$x$" from the shared memory and compute the gradient component $i$ using "$x$":
          $d_i \leftarrow [\nabla f(x)]_i;$
    • Update "$x$" in the shared memory by the proximal operation, performed atomically:
          $x_i \leftarrow \mathcal{P}_{(\gamma/L_{\max})\lambda\Omega_i}\left(x_i - \frac{\gamma}{L_{\max}} d_i\right).$
**until** whenever.
```

Global View of AsySPCD

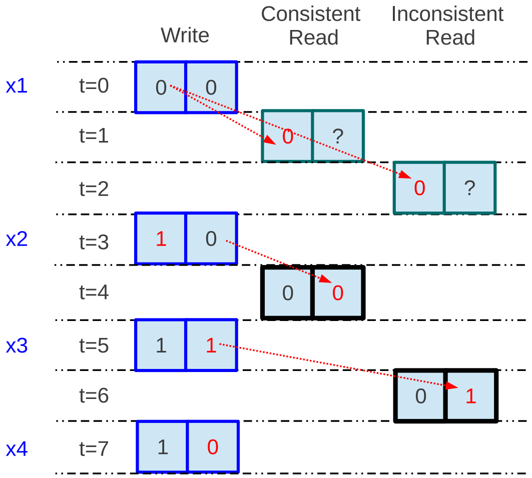
Global counter k incremented when one of the cores makes an update:

```
• Choose $x^0$, steplength parameter $\gamma > 0$;
for $k = 0, 1, 2, \dots$ do
    • Choose $i_k \in \{1, 2, \cdots, n\}$ uniformly at random;
    • Read components of $x$ from shared memory needed to compute
    $[\nabla f(x)]_{i_k}$, denoting the local version of $x$ by $\hat{x}^k$;
    • Update component $i_k$ of $x$ (atomically):
        $x_{i_k}^{k+1} \leftarrow \mathcal{P}_{(\gamma/L_{\max})\lambda\Omega_{i_k}} \left( x_{i_k}^k - \frac{\gamma}{L_{\max}} [\nabla f(\hat{x}^k)]_{i_k} \right).$
end for
```

Vector $\hat{x}^k$ may never appear in shared memory at any point in time. The elements of x may have been updated repeatedly during reading of $\hat{x}^k$, which means that the components of $\hat{x}^k$ may have different "ages."

We call this phenomenon **inconsistent read**.

Consistent Read vs. Inconsistent Read



Consistent / Inconsistent Read

Expressing Read-Inconsistency

Difference between $\hat{x}^k$ and x^k is expressed in terms of "missed updates:"

$$x^k = \hat{x}^k + \sum_{t \in J(k)} (x^{t+1} - x^t)$$

where $J(k)$ defines the iterate set of updates missed in reading $\hat{x}^k$.

We assume τ to be the upper bound of ages of all elements in $J(k)$:

$$\tau \geq k - \min\{t \mid t \in J(k)\}.$$

Example: our assumptions would be satisfied with $\tau = 10$ when

$$x^{100} = \hat{x}^{100} + \sum_{t \in \{91,95,98,99\}} (x^{t+1} - x^t)$$

τ is related strongly to the number of cores / processors that can be used in the computation. The number of updates we would expect to miss between reading and updating x is similar to the number of cores.

Notation

Recall that

- $L_{\max}$: component Lipschitz constant ("max diagonal of Hessian")

$$|[\nabla f(x + te_i)]_i - [\nabla f(x)]_i| \le L_{\max}|t| \quad \forall x, t, i;$$

- L_{res}: restricted Lipschitz constant ("max row-norm of Hessian")

$$\|\nabla f(x + te_i) - \nabla f(x)\|_2 \le L_{\text{res}}|t| \quad \forall x, t, i;$$

- $\Lambda := L_{\text{res}}/L_{\max}$ measures the degree of diagonal dominance.
    - 1 for separable f,
    - 2 for convex quadratic f with diagonally dominant Hessian,
    - $\sqrt{n}$ for general convex quadratic.
- S: the solution set of (1);

Key to Analysis

Recall iteration:

$$x_{i_k}^{k+1} = \mathcal{P}_{(\gamma/L_{\max})\lambda\Omega_{i_k}} \left(x_{i_k}^k - \frac{\gamma}{L_{\max}} [\nabla f(\hat{x}^k)]_{i_k} \right).$$

Choose some $\rho > 1$ and choose γ so that

$$\mathbb{E}(\|x^k - x^{k-1}\|^2) \leq \rho \mathbb{E}(\|x^{k+1} - x^k\|^2) \quad \text{“}\rho\text{-condition”}.$$

Not too much change in the step at each iteration
$\Rightarrow$ not too much change in the gradient
$\Rightarrow$ not too much price to pay for using outdated information.

Want to choose γ small enough to satisfy this property but large enough to get steady convergence.

Strike a balance, as in asynchronous randomized Kaczmarz.

Main Assumption: Optimal Strong Convexity (OSC)

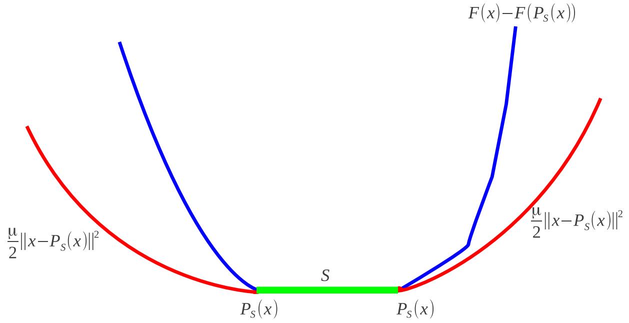
Optimal strong convexity parameter $\sigma > 0$

$$F(x) - F(\mathcal{P}_S(x)) \geq \frac{\sigma}{2}\|x - \mathcal{P}_S(x)\|^2, \quad \text{for all } x \in \text{dom} F.$$

Weaker than usual strong convexity — allows nonunique solutions.

- $F(x) = f(Ax)$ with strongly convex f.
- Squared hinge loss: $F(x) = \sum_k \max(a_k^T x - b_k,\ 0)^2$;

An OSC (but not strongly convex) function:


$F(x) - F(P_S(x))$
$\frac{\mu}{2}\|x - P_S(x)\|^2$
$\frac{\mu}{2}\|x - P_S(x)\|^2$
S
$P_S(x)$
$P_S(x)$

Main Theorem: OSC yields a Linear Rate

Theorem

For any $\rho > 1 + 4/\sqrt{n}$*, define*

$$\theta := \frac{\rho^{(\tau+1)/2} - \rho^{1/2}}{\rho^{1/2} - 1} \quad \theta' := \frac{\rho^{(\tau+1)} - \rho}{\rho - 1} \quad \psi := 1 + \frac{\tau\theta'}{n} + \frac{\Lambda\theta}{\sqrt{n}}.$$

and choose

$$\gamma \leq \min\left\{\frac{1}{\psi}, \frac{\sqrt{n}(1-\rho^{-1}) - 4}{4(1+\theta)\Lambda}\right\}.$$

Then the "ρ*-condition" is satisfied at all* j*, and we have*

$$\begin{aligned}&\mathbb{E}\|x^k - \mathcal{P}_S(x^k)\|^2 + 2\gamma(\mathbb{E}F(x^k) - F^*)\\ &\quad\leq \left(1 - \frac{\sigma}{n(l+\gamma^{-1})}\right)^k \left(\|x^0 - \mathcal{P}_S(x^0)\|^2 + 2\gamma(F(x^0) - F^*)\right).\end{aligned}$$

Notes on the Result

Rate depends intuitively on the various quantities involved:

- Smaller $\gamma \Rightarrow$ slower rate;
- Smaller $\sigma \Rightarrow$ slower rate;
- Larger $\Lambda = L_{\text{res}}/L_{\max}$ implies smaller γ and thus slower rate.
- Larger delay $\tau \Rightarrow$ slower rate.

Dependence on ρ is a bit more complicated, but best to choose ρ near its lower bound.

Special Case

Corollary

Consider the regime in which τ satisfies

$$4e\Lambda(\tau+1)^2 \le \sqrt{n},$$

and define

$$\rho = \left(1 + \frac{4e\Lambda(\tau+1)}{\sqrt{n}}\right)^2.$$

Thus we can choose $\gamma = \frac{1}{2}$, and the rate simplifies to:

$$\mathbb{E}(F(x^k) - F^*) \le \left(1 - \frac{\sigma}{n(l + 2L_{\max})}\right)^k (L_{\max}\|x^0 - \mathcal{P}_S(x^0)\|^2 + F(x^0) - F^*).$$

If the diagonal dominance properties are good ($\Lambda \sim 1$) we have $\tau \sim n^{1/4}$.

In earlier work, with consistent read and no regularization, get $\tau \sim n^{1/2}$.

General Convex (without OSC): Sublinear Rate

Theorem

Define ψ and γ as in the main theorem, have

$$\mathbb{E}(F(x^k) - F^*) \le \frac{n(L_{\max}\gamma^{-1}\|x^0 - \mathcal{P}_S(x^0)\|^2 + 2(F(x^0) - F^*))}{2(k+n)}.$$

Roughly "$1/k$" behavior (sublinear rate).

Corollary

Assuming $4e\Lambda(\tau+1)^2 \le \sqrt{n}$ and setting ρ and $\gamma = 1/2$ as above, we have

$$\mathbb{E}(F(x^k) - F^*) \le \frac{n(L_{\max}\|x^0 - \mathcal{P}_S(x^0)\|^2 + F(x^0) - F^*)}{k+n}.$$

Computational Experiments

Implemented on a 40-core Intel Xeon, containing 4 sockets $\times$ 10 cores.

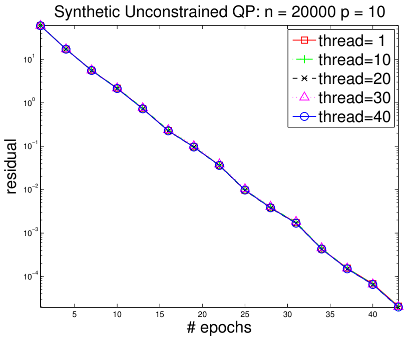
We do sampling without replacement: each thread/core is assign a subset of gradient components, and sweeps through these in order. Order is shuffled periodically (once per epoch, or less frequently).

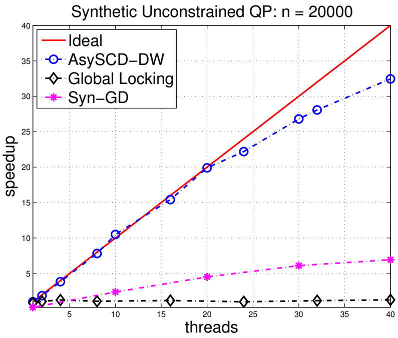
This departs from the analyzed version, which assumes that sampling with replacement for indices i_k.

Unconstrained: 4-socket, 40-core Intel Xeon

$$\min_x \quad \|Ax - b\|^2 + 0.5\|x\|^2$$

where $A \in \mathbb{R}^{m\times n}$ is a Gaussian random matrix ($m = 6000$, $n = 20000$, data size≈3GB, columns are normalized to 1). $\Lambda \approx 2.2$. Choose $\gamma = 1$. Takes 3-4 seconds to achieve the accuracy 10^{-5} on 40 cores.

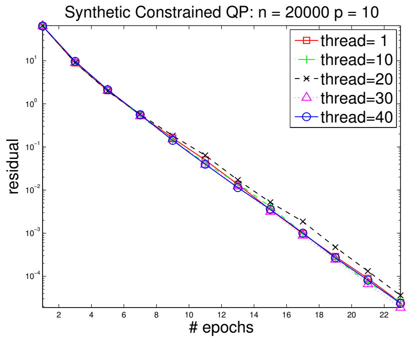


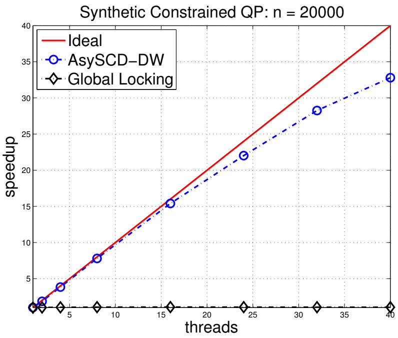


Constrained: 4-socket, 40-core Intel Xeon

$$\min_{x \geq 0} \quad (x-z)^T (A^T A + 0.5I)(x-z) \quad ,$$

where $A \in \mathbb{R}^{m \times n}$ is a Gaussian random matrix ($m = 6000$, $n = 20000$, columns are normalized to 1) and z is a Gaussian random vector. $L_{\text{res}}/L_{\max} \approx 2.2$. Choose $\gamma = 1$.

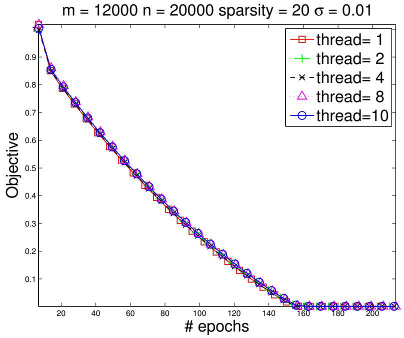


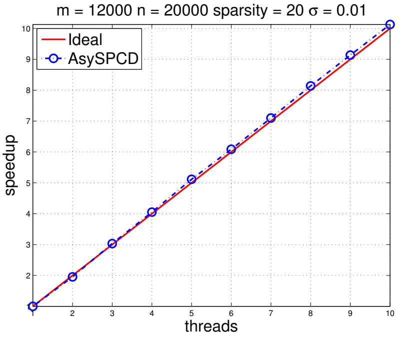


Experiments: 1-socket, 10-core Intel Xeon

$$\min_x \quad \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1,$$

where $A \in \mathbb{R}^{m \times n}$ is a Gaussian random matrix ($m = 12000$, $n = 20000$, data size≈3GB),$b = A * \text{sprandn}(n, 1, 20) + 0.01 * \text{randn}(n, 1)$, and $\lambda = 0.2\sqrt{m \log(n)}$. $L_{\text{res}}/L_{\max} \approx 2.2$. Choose $\gamma = 1$.





Block Coordinate Descent (BCD)

A common and useful extension is to update blocks of coordinates, rather than single coordinates.

At iteration k, select a subset $\mathcal{I}_k \subset \{1, 2, \dots, n\}$ and update just the components in $\mathcal{I}_k$.

As for basic coordinate descent, there are different techniques for choosing $\mathcal{I}_k$ (random, cyclic, essentially cyclic), and different strategies for updating the $\mathcal{I}_k$ components of x.

Can do regularized block coordinate descent provided that the regularizer $\Omega(x)$ is separable with respect to $\mathcal{I}_k$, that is,

$$\Omega(x) = \Omega_{\mathcal{I}_k}(x_{\mathcal{I}_k}) + \Omega_{\mathcal{I}_k^c}(x_{\mathcal{I}_k^c}).$$

Fairly straightforward to generalize the analysis of basic CD.

BCD also opens a path to synchronous parallel implementation, e.g. the components within $\mathcal{I}_k$ can be updated simultaneously, in parallel.

Synchronous Parallel Algorithms

Numerous recent papers on parallel synchronous methods. Examples:

- [Richtarik and Takac, 2013]: separable regularization, internal parallelism in updating block $\mathcal{I}_k$ at iteration k.
- [Marecek et al., 2014]: Partition $\{1, 2, \dots, n\}$ between processors. Two levels of parallelism.
- [Jaggi et al., 2014]: ERM model (encompasses SVM, least squares) with $\{1, 2, \dots, n\}$ partitioned between processors. Nonlinear block Gauss-Jacobi.

Alternative Analysis: “Totally” Asynchronous

A much earlier and much shorter analysis of asynchronous CD appears in [Bertsekas and Tsitsiklis, 1989]. It allows total asynchronicity — no bound on the delay between reading and updating x — no need for finite τ. Need only assume that each component of x is updated infinitely often.

Assume f convex, smooth and that the mapping $T : \mathbb{R}^n \to \mathbb{R}^n$ defined by $T(x) := x - \alpha \nabla f(x)$ for some $\alpha > 0$ is ℓ_∞-norm contractive, that is,

$$\|T(x) - x^*\|_\infty \le \eta \|x - x^*\|_\infty \quad \text{for some } \eta \in (0,1).$$

Then with $\alpha_k \equiv \alpha$ in the totally asynchronous randomized CD algorithm, we have $x^k \to x^*$.

Powerful result. But the contraction assumption can fail even for strongly convex functions: e.g. $f(x) = (1/2)x^T Qx$ with

$$Q = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

Accelerated Variants

Accelerated first-order methods [Nesterov, 1983] have also become highly popular in recent years; the revival traces to [Nesterov, 2004] and [Beck and Teboulle, 2009].

They maintain multiple sequences of vectors $\{x^k\}$, $\{v^k\}$, $\{y^k\}$. Steps are combinations of all gradients encountered so far, not just the latest one. Convergence rates are faster e.g. $O(1/k^2)$ instead of $O(1/k)$.

[Nesterov, 2009] describes an accelerated methods that uses just a subvector of the gradient $\nabla f(x^j)$. But is has the apparent drawback of having to manipulate dense vectors, so at least $O(n)$ operations per iteration are required.

Recent work has shown that this approach can be implemented efficiently in some cases! An early idea for accelerated Kaczmarz is [Liu et al., 2013b], but [Lee and Sidford, 2013] have a much nicer idea that uses a clever change of variables. See also [Fercoq and Richtarik, 2013], [Lin et al., 2014].

Conclusions

- There is a renewed interest in coordinate descent methods, driven by applications (including machine learning).
- CD methods are a good match to parallel (multicore) computer architectures.
- We can analyze asynchronous parallel algorithms, with a computing model and results that approximate reality pretty well.

[Liu et al., 2013a], [Liu et al., 2014], [Liu and Wright, 2014].

References I

Beck, A. and Teboulle, M. (2009).
A fast iterative shrinkage-threshold algorithm for linear inverse problems.
SIAM Journal on Imaging Sciences, 2(1):183–202.

Beck, A. and Tetruashvili, L. (2013).
On the convergence of block coordinate descent methods.
SIAM Journal on Optimization, 23(4):2037–2060.

Bertsekas, D. P. and Tsitsiklis, J. N. (1989).
Parallel and Distributed Computation: Numerical Methods.
Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Fercoq, O. and Richtarik, P. (2013).
Accelerated, parallel, and proximal coordinate descent.
Technical Report arXiv:1312.5799, School of Mathematics, University of Edinburgh.

Frommer, A. and Szyld, D. (2000).
On asynchronous iterations.
Journal of Computational and Applied Mathematics, 123:201–216.

Jaggi, M., Smith, V., Takác, M., Terhorst, J., Krishnan, S., Hoffman, T., and Jordan, M. I. (2014).
Communication-efficient distributed dual coordinate ascent.
Advances in Neural Information Processing Systems, 27.

References II

Lee, Y. T. and Sidford, A. (2013).
Efficient accelerated coordinate descent methods and faster algorihtms for solving linear systems.
In *54th Annual Symposium on Foundations of Computer Science*, pages 147–156.

Lin, Q., Lu, Z., and Xiao, L. (2014).
An accelerated proximal coordinate gradient method and its application to empirical risk minimization.
Technical Report arXiv:1407.1296, Microsoft Research.

Liu, J. and Wright, S. J. (2014).
Asynchronous stochastic coordinate descent: Parallelism and convergence properties.
Technical Report arXiv:1403.3862, University of Wisconsin, Madison.
To appear in *SIAM Journal on Optimization*.

Liu, J., Wright, S. J., Ré, C., Bittorf, V., and Sridhar, S. (2013a).
An asynchronous parallel stochastic coordinate descent algorithm.
Technical Report arXiv:1311.1873, Computer Sciences Department, University of Wisconsin-Madison.
To appear in *Journal of Machine Learning Research*.

References III

Liu, J., Wright, S. J., and Sridhar, S. (2013b).
An accelerated randomized Kaczmarz algorithm.
Technical Report arXiv 1310.2887, Computer Sciences Department, University of Wisconsin-Madison.
To appear in *Mathematics of Computation*.

Liu, J., Wright, S. J., and Sridhar, S. (2014).
An asynchronous parallel randomized Kaczmarz algorithm.
Technical Report arXiv:1401.4780, Computer Sciences Department, University of Wisconsin-Madison.

Marecek, J., Richtarik, P., and Takac, M. (2014).
Distributed block coordinate descent for minimizing partially separable functions.
Technical Report arXiv:1406.0238.

Nesterov, Y. (1983).
A method for unconstrained convex problem with the rate of convergence $o(1/k^2)$.
Doklady AN SSSR, 269:543–547.

Nesterov, Y. (2004).
Introductory Lectures on Convex Optimization: A Basic Course.
Kluwer Academic Publishers.

References IV

Nesterov, Y. (2009).
Primal-dual subgradient methods for convex programs.
Mathematical Programming, Series B, 120:221–259.

Niu, F., Recht, B., Ré, C., and Wright, S. J. (2011).
HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent.
Technical report, University of Wisconsin-Madison.

Powell, M. J. D. (1973).
On search directions for minimization algorithms.
Mathematical Programming, 4:193–201.

Richtarik, P. and Takac, M. (2013).
Parallel coordinate descent methods for big data optimization.
Technical Report arXiv:1212.0873, School of Mathematics, University of Edinburgh.

Strohmer, T. and Vershynin, R. (2009).
A randomized Kaczmarz algorithm with exponential convergence.
Journal of Fourier Analysis and Applications, 15:262–278.