Introduction to large-scale optimization

(Lecture 3)

Suvrit Sra

Massachusetts Institute of Technology

Microsoft Research India
Machine Learning Summer School, June 2015

Course materials

- http://suvrit.de/teach/msr2015/
- Some references:
  - *Introductory lectures on convex optimization* – Nesterov
  - *Convex optimization* – Boyd & Vandenberghe
  - *Nonlinear programming* – Bertsekas
  - *Convex Analysis* – Rockafellar
  - *Fundamentals of convex analysis* – Urruty, Lemaréchal
  - *Lectures on modern convex optimization* – Nemirovski
  - *Optimization for Machine Learning* – Sra, Nowozin, Wright
- Some related courses:
  - EE227A, Spring 2013, (UC Berkeley)
  - 10-801, Spring 2014 (CMU)
  - EE364a,b (Boyd, Stanford)
  - EE236b,c (Vandenberghe, UCLA)
- NIPS, ICML, UAI, AISTATS, SIOPT, Math. Prog.

Outline

- Recap on convexity
- Recap on duality, optimality
- First-order optimization algorithms
- Proximal methods, operator splitting
- Incremental methods, stochastic gradient
- High-level view of parallel, distributed

Large-scale ML

Regularized Empirical Risk Minimization

$$\min_{w} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^T x_i) + \lambda r(w).$$

This is the $f(w) + r(w)$ “composite objective” form we saw.

(e.g., regression, logistic regression, lasso, CRFs, etc.)

Large-scale ML

Regularized Empirical Risk Minimization

$$\min_{w} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^T x_i) + \lambda r(w).$$

This is the $f(w) + r(w)$ “composite objective” form we saw.

(e.g., regression, logistic regression, lasso, CRFs, etc.)

- training data: $(x_i, y_i) \in \mathbb{R}^d \times \mathcal{Y}$ (i.i.d.)
- large-scale ML: Both d and n are large:
  - d: dimension of each input sample
  - n: number of training data points / samples
- Assume training data “sparse”; so total datasize $\ll dn$.
- Running time $O(\#\text{nnz})$

Regularized Risk Minimization

Training cost $\widehat{F}(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^T x_i) + \lambda r(w)$

Generalization $F(w) = \mathbb{E}_{(x,y)}[\ell(y, w^T x)] + \lambda r(w)$

Regularized Risk Minimization

Training cost $\widehat{F}(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^T x_i) + \lambda r(w)$

Generalization $F(w) = \mathbb{E}_{(x,y)}[\ell(y, w^T x)] + \lambda r(w)$

Single pass through data for $F(w)$ by sampling n points

Multiple passes if only minimizing empirical cost $\widehat{F}(w)$

Stochastic optimization

$$\min_{x\in\mathcal{X}} F(x) := \mathbb{E}_{\xi}[f(x,\xi)]$$

(f: loss; x: parameters; ξ: data samples)

Setup

1. $\mathcal{X} \subset \mathbb{R}^d$ compact convex set

Stochastic optimization

$$\min_{x \in \mathcal{X}} F(x) := \mathbb{E}_{\xi}[f(x, \xi)]$$
(f: loss; x: parameters; ξ: data samples)

Setup

1. $\mathcal{X} \subset \mathbb{R}^d$ compact convex set

2. ξ r.v. with distribution P on $\Omega \subset \mathbb{R}^d$

Stochastic optimization

$$\min_{x\in\mathcal{X}} F(x) := \mathbb{E}_\xi[f(x,\xi)]$$
(f: loss; x: parameters; ξ: data samples)

Setup

1. $\mathcal{X} \subset \mathbb{R}^d$ compact convex set

2. ξ r.v. with distribution P on $\Omega \subset \mathbb{R}^d$

3. The expectation

$$\mathbb{E}_\xi[f(x,\xi)] = \int_\Omega f(x,\xi)dP(\xi)$$

is well-defined and finite valued for every $x \in \mathcal{X}$.

Stochastic optimization

$$\min_{x \in \mathcal{X}} F(x) := \mathbb{E}_{\xi}[f(x,\xi)]$$

(f: loss; x: parameters; ξ: data samples)

Setup

1. $\mathcal{X} \subset \mathbb{R}^d$ compact convex set

2. ξ r.v. with distribution P on $\Omega \subset \mathbb{R}^d$

3. The expectation

$$\mathbb{E}_{\xi}[f(x,\xi)] = \int_{\Omega} f(x,\xi) dP(\xi)$$

is well-defined and finite valued for every $x \in \mathcal{X}$.

4. For every $\xi \in \Omega$, $f(\cdot,\xi)$ is convex

Stochastic optimization

Assumption 1: Possible to generate iid samples $\xi_1, \xi_2, \ldots$

Assumption 2: Oracle yields stochastic gradient $g(x,\xi)$, i.e.,

$$G(x) := \mathbb{E}[g(x,\xi)] \quad \text{s.t.} \quad G(x) \in \partial F(x).$$

Stochastic optimization

Assumption 1: Possible to generate iid samples $\xi_1, \xi_2, \ldots$
Assumption 2: Oracle yields stochastic gradient $g(x, \xi)$, i.e.,

$$G(x) := \mathbb{E}[g(x, \xi)] \quad \text{s.t.} \quad G(x) \in \partial F(x).$$

Theorem Let $\xi \in \Omega$; If $f(\cdot, \xi)$ is convex, and $F(\cdot)$ is finite valued in a neighborhood of x, then

$$\partial F(x) = \mathbb{E}[\partial_x f(x, \xi)].$$

Stochastic optimization

Assumption 1: Possible to generate iid samples $\xi_1, \xi_2, \ldots$
Assumption 2: Oracle yields stochastic gradient $g(x, \xi)$, i.e.,

$$G(x) := \mathbb{E}[g(x, \xi)] \quad \text{s.t.} \quad G(x) \in \partial F(x).$$

Theorem Let $\xi \in \Omega$; If $f(\cdot, \xi)$ is convex, and $F(\cdot)$ is finite valued in a neighborhood of x, then

$$\partial F(x) = \mathbb{E}[\partial_x f(x, \xi)].$$

- So $g(x, \omega) \in \partial_x f(x, \omega)$ is a stochastic subgradient.

Stochastic optimization – solving

- ♣ Stochastic Approximation (SA) / Stochastic gradient (SGD)
  - ▶ Sample ξ_k iid

Stochastic optimization – solving

♣ Stochastic Approximation (SA) / Stochastic gradient (SGD)

- Sample ξ_k iid
- Generate stochastic subgradient $g(x, \xi)$

Stochastic optimization – solving

♣ Stochastic Approximation (SA) / Stochastic gradient (SGD)

- Sample ξ_k iid
- Generate stochastic subgradient $g(x, \xi)$
- Use that in a subgradient method

Stochastic optimization – solving

- ♣ Stochastic Approximation (SA) / Stochastic gradient (SGD)
  - ▶ Sample ξ_k iid
  - ▶ Generate stochastic subgradient $g(x, \xi)$
  - ▶ Use that in a subgradient method
- ♣ Sample average approximation (SAA)

Stochastic optimization – solving

- ♣ Stochastic Approximation (SA) / Stochastic gradient (SGD)
  - ▶ Sample ξ_k iid
  - ▶ Generate stochastic subgradient $g(x, \xi)$
  - ▶ Use that in a subgradient method
- ♣ Sample average approximation (SAA)
  - ▶ Generate n iid samples, $\xi_1, \ldots, \xi_n$

Stochastic optimization – solving

- ♣ Stochastic Approximation (SA) / Stochastic gradient (SGD)
  - ▶ Sample ξ_k iid
  - ▶ Generate stochastic subgradient $g(x, \xi)$
  - ▶ Use that in a subgradient method
- ♣ Sample average approximation (SAA)
  - ▶ Generate n iid samples, $\xi_1, \ldots, \xi_n$
  - ▶ Consider **empirical objective** $\hat{F}_n := n^{-1} \sum_i f(x, \xi_i)$

Stochastic optimization – solving

- ♣ Stochastic Approximation (SA) / Stochastic gradient (SGD)
  - ▶ Sample ξ_k iid
  - ▶ Generate stochastic subgradient $g(x, \xi)$
  - ▶ Use that in a subgradient method
- ♣ Sample average approximation (SAA)
  - ▶ Generate n iid samples, $\xi_1, \ldots, \xi_n$
  - ▶ Consider **empirical objective** $\hat{F}_n := n^{-1} \sum_i f(x, \xi_i)$
  - ▶ SAA refers to creation of this **sample average problem**
  - ▶ Minimizing $\hat{F}_n$ still needs to be done!

Stochastic gradient

SA or stochastic (sub)-gradient

- Let $x_0 \in \mathcal{X}$
- For $k \geq 0$
  - Sample ξ_k; compute $g(x_k, \xi_k)$ using oracle
  - Update $x_{k+1} = P_{\mathcal{X}}(x_k - \alpha_k g(x_k, \xi_k))$, where $\alpha_k > 0$

Stochastic gradient

SA or stochastic (sub)-gradient

- Let $x_0 \in \mathcal{X}$
- For $k \geq 0$
  - Sample ξ_k; compute $g(x_k, \xi_k)$ using oracle
  - Update $x_{k+1} = P_{\mathcal{X}}(x_k - \alpha_k g(x_k, \xi_k))$, where $\alpha_k > 0$

We'll simply write

$$\boxed{x_{k+1} = P_{\mathcal{X}}\big(x_k - \alpha_k g_k\big)}$$

Stochastic gradient

SA or stochastic (sub)-gradient

- Let $x_0 \in \mathcal{X}$
- For $k \geq 0$
  - Sample ξ_k; compute $g(x_k, \xi_k)$ using oracle
  - Update $x_{k+1} = P_{\mathcal{X}}(x_k - \alpha_k g(x_k, \xi_k))$, where $\alpha_k > 0$

We'll simply write

$$\boxed{x_{k+1} = P_{\mathcal{X}}\big(x_k - \alpha_k g_k\big)}$$

Does this work?

Convergence Analysis

- x_k depends on rvs $\xi_1, \ldots, \xi_{k-1}$, so itself random

Convergence Analysis

- x_k depends on rvs $\xi_1, \ldots, \xi_{k-1}$, so itself random
- Of course, x_k does not depend on ξ_k

Convergence Analysis

- x_k depends on rvs $\xi_1, \ldots, \xi_{k-1}$, so itself random
- Of course, x_k does not depend on ξ_k
- Subgradient method analysis hinges upon: $\|x_k - x^*\|^2$

Convergence Analysis

- x_k depends on rvs $\xi_1, \ldots, \xi_{k-1}$, so itself random
- Of course, x_k does not depend on ξ_k
- Subgradient method analysis hinges upon: $\|x_k - x^*\|^2$
- Stochastic subgradient hinges upon: $\mathbb{E}[\|x_k - x^*\|^2]$

Convergence Analysis

- x_k depends on rvs $\xi_1, \ldots, \xi_{k-1}$, so itself random
- Of course, x_k does not depend on ξ_k
- Subgradient method analysis hinges upon: $\|x_k - x^*\|^2$
- Stochastic subgradient hinges upon: $\mathbb{E}[\|x_k - x^*\|^2]$

Denote: $R_k := \|x_k - x^*\|^2$ and $r_k := \mathbb{E}[R_k] = \mathbb{E}[\|x_k - x^*\|^2]$

Convergence Analysis

- x_k depends on rvs $\xi_1, \ldots, \xi_{k-1}$, so itself random
- Of course, x_k does not depend on ξ_k
- Subgradient method analysis hinges upon: $\|x_k - x^*\|^2$
- Stochastic subgradient hinges upon: $\mathbb{E}[\|x_k - x^*\|^2]$

Denote: $R_k := \|x_k - x^*\|^2$ and $r_k := \mathbb{E}[R_k] = \mathbb{E}[\|x_k - x^*\|^2]$

Bounding R_{k+1}

$$R_{k+1} \quad = \quad \|x_{k+1} - x^*\|_2^2 = \|P_{\mathcal{X}}(x_k - \alpha_k g_k) - P_{\mathcal{X}}(x^*)\|_2^2$$

Convergence Analysis

- x_k depends on rvs $\xi_1, \ldots, \xi_{k-1}$, so itself random
- Of course, x_k does not depend on ξ_k
- Subgradient method analysis hinges upon: $\|x_k - x^*\|^2$
- Stochastic subgradient hinges upon: $\mathbb{E}[\|x_k - x^*\|^2]$

Denote: $R_k := \|x_k - x^*\|^2$ and $r_k := \mathbb{E}[R_k] = \mathbb{E}[\|x_k - x^*\|^2]$

Bounding R_{k+1}

$$\begin{aligned} R_{k+1} &= \|x_{k+1} - x^*\|_2^2 = \|P_{\mathcal{X}}(x_k - \alpha_k g_k) - P_{\mathcal{X}}(x^*)\|_2^2 \\ &\leq \|x_k - x^* - \alpha_k g_k\|_2^2 \end{aligned}$$

Convergence Analysis

- x_k depends on rvs $\xi_1, \ldots, \xi_{k-1}$, so itself random
- Of course, x_k does not depend on ξ_k
- Subgradient method analysis hinges upon: $\|x_k - x^*\|^2$
- Stochastic subgradient hinges upon: $\mathbb{E}[\|x_k - x^*\|^2]$

Denote: $R_k := \|x_k - x^*\|^2$ and $r_k := \mathbb{E}[R_k] = \mathbb{E}[\|x_k - x^*\|^2]$

Bounding R_{k+1}

$$\begin{aligned} R_{k+1} &= \|x_{k+1} - x^*\|_2^2 = \|P_{\mathcal{X}}(x_k - \alpha_k g_k) - P_{\mathcal{X}}(x^*)\|_2^2 \\ &\leq \|x_k - x^* - \alpha_k g_k\|_2^2 \\ &= R_k + \alpha_k^2 \|g_k\|_2^2 - 2\alpha_k \langle g_k, x_k - x^* \rangle. \end{aligned}$$

Convergence analysis

$$\boxed{R_{k+1} \leq R_k + \alpha_k^2 \|g_k\|_2^2 - 2\alpha_k \langle g_k, x_k - x^* \rangle}$$

Convergence analysis

$$\boxed{R_{k+1} \le R_k + \alpha_k^2 \|g_k\|_2^2 - 2\alpha_k \langle g_k,\, x_k - x^* \rangle}$$

- **Assume:** $\|g_k\|_2 \le M$ on $\mathcal{X}$
- Taking expectation:

$$r_{k+1} \le r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle g_k,\, x_k - x^* \rangle].$$

Convergence analysis

$$\boxed{R_{k+1} \leq R_k + \alpha_k^2 \|g_k\|_2^2 - 2\alpha_k \langle g_k,\, x_k - x^* \rangle}$$

- **Assume:** $\|g_k\|_2 \leq M$ on $\mathcal{X}$
- Taking expectation:

$$r_{k+1} \leq r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle g_k,\, x_k - x^* \rangle].$$

- We need to now get a handle on the last term

Convergence analysis

$$\boxed{R_{k+1} \leq R_k + \alpha_k^2 \|g_k\|_2^2 - 2\alpha_k \langle g_k, x_k - x^* \rangle}$$

- **Assume:** $\|g_k\|_2 \leq M$ on $\mathcal{X}$
- Taking expectation:
$$r_{k+1} \leq r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle g_k, x_k - x^* \rangle].$$
- We need to now get a handle on the last term
- Since x_k is independent of ξ_k, we have

$$\mathbb{E}[\langle x_k - x^*, g(x_k, \xi_k) \rangle] \quad =$$

Convergence analysis

$$\boxed{R_{k+1} \leq R_k + \alpha_k^2 \|g_k\|_2^2 - 2\alpha_k \langle g_k,\, x_k - x^* \rangle}$$

- **Assume:** $\|g_k\|_2 \leq M$ on $\mathcal{X}$
- Taking expectation:

$$r_{k+1} \leq r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle g_k,\, x_k - x^* \rangle].$$

- We need to now get a handle on the last term
- Since x_k is independent of ξ_k, we have

$$\begin{aligned} \mathbb{E}[\langle x_k - x^*,\, g(x_k, \xi_k) \rangle] &= \mathbb{E}\left\{ \mathbb{E}[\langle x_k - x^*,\, g(x_k, \xi_k) \rangle \mid \xi_{[1..(k-1)]}] \right\} \\ &= \end{aligned}$$

Convergence analysis

$$\boxed{R_{k+1} \leq R_k + \alpha_k^2 \|g_k\|_2^2 - 2\alpha_k \langle g_k,\, x_k - x^* \rangle}$$

- **Assume:** $\|g_k\|_2 \leq M$ on $\mathcal{X}$
- Taking expectation:

$$r_{k+1} \leq r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle g_k,\, x_k - x^* \rangle].$$

- We need to now get a handle on the last term
- Since x_k is independent of ξ_k, we have

$$\begin{aligned}
\mathbb{E}[\langle x_k - x^*,\, g(x_k, \xi_k) \rangle] &= \mathbb{E}\left\{\mathbb{E}[\langle x_k - x^*,\, g(x_k, \xi_k) \rangle \mid \xi_{[1..(k-1)]}]\right\} \\
&= \mathbb{E}\left\{\langle x_k - x^*,\, \mathbb{E}[g(x_k, \xi_k) \mid \xi_{[1..(k-1)]}] \rangle\right\} \\
&=
\end{aligned}$$

Convergence analysis

$$\boxed{R_{k+1} \le R_k + \alpha_k^2 \|g_k\|_2^2 - 2\alpha_k \langle g_k,\, x_k - x^* \rangle}$$

- **Assume:** $\|g_k\|_2 \le M$ on $\mathcal{X}$
- Taking expectation:

$$r_{k+1} \le r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle g_k,\, x_k - x^* \rangle].$$

- We need to now get a handle on the last term
- Since x_k is independent of ξ_k, we have

$$\begin{aligned}
\mathbb{E}[\langle x_k - x^*,\, g(x_k, \xi_k)\rangle] &= \mathbb{E}\left\{\mathbb{E}[\langle x_k - x^*,\, g(x_k, \xi_k)\rangle \mid \xi_{[1..(k-1)]}]\right\} \\
&= \mathbb{E}\left\{\langle x_k - x^*,\, \mathbb{E}[g(x_k, \xi_k) \mid \xi_{[1..(k-1)]}]\rangle\right\} \\
&= \mathbb{E}[\langle x_k - x^*,\, G_k\rangle], \quad G_k \in \partial F(x_k).
\end{aligned}$$

Convergence analysis

It remains to bound: $\mathbb{E}[\langle x_k - x^*, G_k \rangle]$

Convergence analysis

$$\boxed{\text{It remains to bound: } \mathbb{E}[\langle x_k - x^*,\, G_k\rangle]}$$

- Since F is cvx, $F(x) \geq F(x_k) + \langle G_k,\, x - x_k\rangle$ for any $x \in \mathcal{X}$.

Convergence analysis

It remains to bound: $\mathbb{E}[\langle x_k - x^*, G_k\rangle]$

- Since F is cvx, $F(x) \geq F(x_k) + \langle G_k, x - x_k\rangle$ for any $x \in \mathcal{X}$.
- Thus, in particular

$$2\alpha_k \mathbb{E}[F(x^*) - F(x_k)] \geq 2\alpha_k \mathbb{E}[\langle G_k, x^* - x_k\rangle]$$

Convergence analysis

It remains to bound: $\mathbb{E}[\langle x_k - x^*, G_k\rangle]$

- Since F is cvx, $F(x) \geq F(x_k) + \langle G_k, x - x_k\rangle$ for any $x \in \mathcal{X}$.
- Thus, in particular

$$2\alpha_k \mathbb{E}[F(x^*) - F(x_k)] \geq 2\alpha_k \mathbb{E}[\langle G_k, x^* - x_k\rangle]$$

Plug this bound back into the r_{k+1} inequality:

$$r_{k+1} \quad \leq \quad r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle G_k, x_k - x^*\rangle]$$

Convergence analysis

$$\boxed{\text{It remains to bound: } \mathbb{E}[\langle x_k - x^*,\, G_k\rangle]}$$

- Since F is cvx, $F(x) \geq F(x_k) + \langle G_k,\, x - x_k\rangle$ for any $x \in \mathcal{X}$.
- Thus, in particular

$$2\alpha_k \mathbb{E}[F(x^*) - F(x_k)] \geq 2\alpha_k \mathbb{E}[\langle G_k,\, x^* - x_k\rangle]$$

Plug this bound back into the r_{k+1} inequality:

$$\begin{aligned} r_{k+1} &\leq r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle G_k,\, x_k - x^*\rangle] \\ 2\alpha_k \mathbb{E}[\langle G_k,\, x_k - x^*\rangle] &\leq r_k - r_{k+1} + \alpha_k M^2 \end{aligned}$$

Convergence analysis

It remains to bound: $\mathbb{E}[\langle x_k - x^*, G_k\rangle]$

- Since F is cvx, $F(x) \geq F(x_k) + \langle G_k, x - x_k\rangle$ for any $x \in \mathcal{X}$.
- Thus, in particular

$$2\alpha_k \mathbb{E}[F(x^*) - F(x_k)] \geq 2\alpha_k \mathbb{E}[\langle G_k, x^* - x_k\rangle]$$

Plug this bound back into the r_{k+1} inequality:

$$\begin{aligned} r_{k+1} &\leq r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle G_k, x_k - x^*\rangle] \\ 2\alpha_k \mathbb{E}[\langle G_k, x_k - x^*\rangle] &\leq r_k - r_{k+1} + \alpha_k M^2 \\ 2\alpha_k \mathbb{E}[F(x_k) - F(x^*)] &\leq r_k - r_{k+1} + \alpha_k M^2. \end{aligned}$$

Convergence analysis

It remains to bound: $\mathbb{E}[\langle x_k - x^*, G_k\rangle]$

- Since F is cvx, $F(x) \geq F(x_k) + \langle G_k, x - x_k\rangle$ for any $x \in \mathcal{X}$.
- Thus, in particular

$$2\alpha_k \mathbb{E}[F(x^*) - F(x_k)] \geq 2\alpha_k \mathbb{E}[\langle G_k, x^* - x_k\rangle]$$

Plug this bound back into the r_{k+1} inequality:

$$\begin{array}{rcl} r_{k+1} & \leq & r_k + \alpha_k^2 M^2 - 2\alpha_k \mathbb{E}[\langle G_k, x_k - x^*\rangle] \\ 2\alpha_k \mathbb{E}[\langle G_k, x_k - x^*\rangle] & \leq & r_k - r_{k+1} + \alpha_k M^2 \\ 2\alpha_k \mathbb{E}[F(x_k) - F(x^*)] & \leq & r_k - r_{k+1} + \alpha_k M^2. \end{array}$$

We've bounded the expected progress; What now?

Convergence analysis

$$2\alpha_k \mathbb{E}[F(x_k) - F(x^*)] \leq r_k - r_{k+1} + \alpha_k M^2.$$

Convergence analysis

$$2\alpha_k \mathbb{E}[F(x_k) - F(x^*)] \le r_k - r_{k+1} + \alpha_k M^2.$$

Sum up over $i = 1, \ldots, k$, to obtain

$$\sum\nolimits_{i=1}^{k} (2\alpha_i \mathbb{E}[F(x_i) - f(x^*)]) \quad \le \quad r_1 - r_{k+1} + M^2 \sum\nolimits_i \alpha_i^2$$

Convergence analysis

$$2\alpha_k \mathbb{E}[F(x_k) - F(x^*)] \leq r_k - r_{k+1} + \alpha_k M^2.$$

Sum up over $i = 1, \ldots, k$, to obtain

$$\begin{aligned} \sum\nolimits_{i=1}^{k} (2\alpha_i \mathbb{E}[F(x_i) - f(x^*)]) &\leq r_1 - r_{k+1} + M^2 \sum\nolimits_i \alpha_i^2 \\ &\leq r_1 + M^2 \sum\nolimits_i \alpha_i^2. \end{aligned}$$

Convergence analysis

$$2\alpha_k \mathbb{E}[F(x_k) - F(x^*)] \le r_k - r_{k+1} + \alpha_k M^2.$$

Sum up over $i = 1, \ldots, k$, to obtain

$$\begin{aligned} \sum\nolimits_{i=1}^{k} (2\alpha_i \mathbb{E}[F(x_i) - f(x^*)]) &\le r_1 - r_{k+1} + M^2 \sum\nolimits_i \alpha_i^2 \\ &\le r_1 + M^2 \sum\nolimits_i \alpha_i^2. \end{aligned}$$

Divide both sides by $\sum_i \alpha_i$, so

Convergence analysis

$$2\alpha_k \mathbb{E}[F(x_k) - F(x^*)] \leq r_k - r_{k+1} + \alpha_k M^2.$$

Sum up over $i = 1, \ldots, k$, to obtain

$$\begin{aligned}\textstyle\sum_{i=1}^{k} (2\alpha_i \mathbb{E}[F(x_i) - f(x^*)]) &\leq r_1 - r_{k+1} + M^2 \textstyle\sum_i \alpha_i^2 \\ &\leq r_1 + M^2 \textstyle\sum_i \alpha_i^2.\end{aligned}$$

Divide both sides by $\sum_i \alpha_i$, so

- Set $\gamma_i = \frac{\alpha_i}{\sum_i^k \alpha_i}$.
- Thus, $\gamma_i \geq 0$ and $\sum_i \gamma_i = 1$

Convergence analysis

$$2\alpha_k \mathbb{E}[F(x_k) - F(x^*)] \leq r_k - r_{k+1} + \alpha_k M^2.$$

Sum up over $i = 1, \ldots, k$, to obtain

$$\begin{aligned} \sum\nolimits_{i=1}^{k} (2\alpha_i \mathbb{E}[F(x_i) - f(x^*)]) &\leq r_1 - r_{k+1} + M^2 \sum\nolimits_i \alpha_i^2 \\ &\leq r_1 + M^2 \sum\nolimits_i \alpha_i^2. \end{aligned}$$

Divide both sides by $\sum_i \alpha_i$, so

- Set $\gamma_i = \frac{\alpha_i}{\sum_i^k \alpha_i}$.
- Thus, $\gamma_i \geq 0$ and $\sum_i \gamma_i = 1$

$$\mathbb{E}\left[\sum\nolimits_i \gamma_i (F(x_i) - F(x^*))\right] \leq \frac{r_1 + M^2 \sum_i \alpha_i^2}{2 \sum_i \alpha_i}$$

Convergence analysis

- But we wish to say something about x_k

Convergence analysis

- But we wish to say something about x_k
- Since $\gamma_i \geq 0$ and $\sum_i^k \gamma_i = 1$, and we have $\gamma_i F(x_i)$

Convergence analysis

- But we wish to say something about x_k
- Since $\gamma_i \geq 0$ and $\sum_i^k \gamma_i = 1$, and we have $\gamma_i F(x_i)$
- Easier to talk about **averaged**

$$\bar{x}_k := \sum_i^k \gamma_i x_i.$$

Convergence analysis

- But we wish to say something about x_k
- Since $\gamma_i \geq 0$ and $\sum_i^k \gamma_i = 1$, and we have $\gamma_i F(x_i)$
- Easier to talk about **averaged**

$$\bar{x}_k := \sum\nolimits_i^k \gamma_i x_i.$$

- $f(\bar{x}_k) \leq \sum_i \gamma_i F(x_i)$ due to convexity

Convergence analysis

- But we wish to say something about x_k
- Since $\gamma_i \geq 0$ and $\sum_i^k \gamma_i = 1$, and we have $\gamma_i F(x_i)$
- Easier to talk about **averaged**

$$\bar{x}_k := \sum\nolimits_i^k \gamma_i x_i.$$

- $f(\bar{x}_k) \leq \sum_i \gamma_i F(x_i)$ due to convexity
- So we finally obtain the inequality

$$\mathbb{E}\big[F(\bar{x}_k) - F(x^*)\big] \leq \frac{r_1 + M^2 \sum_i \alpha_i^2}{2\sum_i \alpha_i}.$$

SGD – finally

♠ Let $D_{\mathcal{X}} := \max_{x \in \mathcal{X}} \|x - x^*\|_2$ (act. only need $\|x_1 - x^*\| \leq D_{\mathcal{X}}$)

♠ Assume $\alpha_i = \alpha$ is a constant. Observe that

$$\mathbb{E}[F(\bar{x}_k) - F(x^*)] \leq \frac{D_{\mathcal{X}}^2 + M^2 k \alpha^2}{2k\alpha}$$

♠ Minimize rhs over $\alpha > 0$; thus $\mathbb{E}[F(\bar{x}_k) - F(x^*)] \leq \frac{D_{\mathcal{X}} M}{\sqrt{k}}$

♠ If k is not fixed in advance, then choose

$$\alpha_i = \frac{\theta D_{\mathcal{X}}}{M\sqrt{i}}, \quad i = 1, 2, \ldots$$

We showed $O(1/\sqrt{k})$ rate

Stochastic optimization – smooth

Theorem Let $f(x,\xi)$ be C_L^1 convex. Let $e_k := \nabla F(x_k) - g_k$ satisfy $\mathbb{E}[e_k] = 0$. Let $\|x_i - x^*\| \le D$. Also, let $\alpha_i = 1/(L+\eta_i)$. Then,

$$\mathbb{E}\left[\sum_{i=1}^{k} F(x_{i+1}) - F(x^*)\right] \le \frac{D^2}{2\alpha_k} + \sum_{i=1}^{k} \frac{\mathbb{E}[\|e_i\|^2]}{2\eta_i}.$$

Stochastic optimization – smooth

Theorem Let $f(x, \xi)$ be C^1_L convex. Let $e_k := \nabla F(x_k) - g_k$ satisfy $\mathbb{E}[e_k] = 0$. Let $\|x_i - x^*\| \leq D$. Also, let $\alpha_i = 1/(L + \eta_i)$. Then,

$$\mathbb{E}\Bigl[\sum\nolimits_{i=1}^{k} F(x_{i+1}) - F(x^*)\Bigr] \leq \frac{D^2}{2\alpha_k} + \sum\nolimits_{i=1}^{k} \frac{\mathbb{E}[\|e_i\|^2]}{2\eta_i}.$$

As before, by using $\bar{x}_k = \frac{1}{k}\sum_{i=1}^{k} x_{i+1}$ we get

$$\mathbb{E}[F(\bar{x}_k) - F(x^*)] \leq \frac{D^2}{2\alpha_k k} + \frac{1}{k}\sum\nolimits_{i=1}^{k} \frac{\mathbb{E}[\|e_i\|^2]}{2\eta_i}.$$

Stochastic optimization – smooth

Theorem Let $f(x,\xi)$ be C^1_L convex. Let $e_k := \nabla F(x_k) - g_k$ satisfy $\mathbb{E}[e_k] = 0$. Let $\|x_i - x^*\| \leq D$. Also, let $\alpha_i = 1/(L+\eta_i)$. Then,

$$\mathbb{E}\Big[\sum\nolimits_{i=1}^{k} F(x_{i+1}) - F(x^*)\Big] \leq \frac{D^2}{2\alpha_k} + \sum\nolimits_{i=1}^{k} \frac{\mathbb{E}[\|e_i\|^2]}{2\eta_i}.$$

As before, by using $\bar{x}_k = \frac{1}{k}\sum_{i=1}^{k} x_{i+1}$ we get

$$\mathbb{E}[F(\bar{x}_k) - F(x^*)] \leq \frac{D^2}{2\alpha_k k} + \frac{1}{k}\sum\nolimits_{i=1}^{k} \frac{\mathbb{E}[\|e_i\|^2]}{2\eta_i}.$$

- Using $\alpha_i = L + \eta_i$ where $\eta_i \propto 1/\sqrt{i}$ we obtain

$$\mathbb{E}[F(\bar{x}_k) - F(x^*)] = O(\tfrac{LD^2}{k}) + O(\tfrac{\sigma D}{\sqrt{k}})$$

where σ bounds the variance $\mathbb{E}[\|e_i\|^2]$

Minimax optimal rate

Stochastic optimization – strongly convex

Theorem Suppose $f(x, \xi)$ are convex and $F(x)$ is μ-strongly convex. Let $\bar{x}_k := \sum_{i=0}^{k-1} \theta_i x_i$, where $\theta_i = \frac{2(i+1)}{(k+1)(k+2)}$, we obtain

$$\mathbb{E}[F(\bar{x}_k) - F(x^*)] \leq \frac{2M^2}{\mu(k+1)}.$$

(*Lacoste-Julien, Schmidt, Bach (2012)*)

With uniform averaging $\bar{x}_k = \frac{1}{k}\sum_i x_i$, we get $O(\log k/k)$.

Extensions

- Proximal stochastic gradient

  $$x_{k+1} = \mathrm{prox}_{\alpha_k h}[x_k - \alpha_k g(x_k, \xi_k)]$$

  (*Xiao 2010; Hu et al. 2009*)

  Accelerated versions also possible (*Ghadimi, Lan (2013)*)

- Related methods:
  - Regularized dual averaging (Nesterov, 2009; Xiao 2010)
  - Stochastic mirror-prox (Nemirovski et al. 2009)
- ...

SAA / Batch problem

$$\boxed{\min F(x) = \mathbb{E}[f(x,\xi)]}$$

Sample Average Approximation (SAA):

- Collect samples $\xi_1, \ldots, \xi_n$
- Empirical objective: $\widehat{F}(x) := \frac{1}{n}\sum_{i=1}^{n} f(x,\xi_i)$
- aka *Empirical Risk Minimization*

SAA / Batch problem

$$\boxed{\min F(x) = \mathbb{E}[f(x,\xi)]}$$

Sample Average Approximation (SAA):

- Collect samples $\xi_1, \ldots, \xi_n$
- Empirical objective: $\widehat{F}(x) := \frac{1}{n}\sum_{i=1}^{n} f(x,\xi_i)$
- aka *Empirical Risk Minimization*
- **Note:** we often optimize $\widehat{F}$ using stochastic subgradient; but theoretical guarantees are then only on the *empirical* suboptimality $E[\widehat{F}(\bar{x}_k)] \le \ldots$

SAA / Batch problem

$$\boxed{\min F(x) = \mathbb{E}[f(x,\xi)]}$$

Sample Average Approximation (SAA):

- Collect samples $\xi_1, \ldots, \xi_n$
- Empirical objective: $\widehat{F}(x) := \frac{1}{n}\sum_{i=1}^{n} f(x,\xi_i)$
- aka *Empirical Risk Minimization*
- **Note:** we often optimize $\widehat{F}$ using stochastic subgradient; but theoretical guarantees are then only on the *empirical* suboptimality $E[\widehat{F}(\bar{x}_k)] \leq \ldots$
- For guarantees on $F(\bar{x}_k)$ more work (*regularization* + concentration)

Finite-sum problems

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

Finite-sum problems

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

Gradient / subgradient methods

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k \nabla f(x_k) \\ x_{k+1} &= x_k - \alpha_k g(x_k), \quad g \in \partial f(x_k) \\ x_{k+1} &= \operatorname{prox}_{\alpha_k r}(x_k - \alpha_k \nabla f(x_k)) \end{aligned}$$

Stochastic gradient

At iteration k, we randomly pick an integer $i(k) \in \{1, 2, \ldots, m\}$

$$x_{k+1} = x_k - \alpha_k \nabla f_{i(k)}(x_k)$$

- The update requires only gradient for $f_{i(k)}$
- Uses unbiased estimate $\mathbb{E}[\nabla f_{i(k)}] = \nabla f$
- One iteration now n times faster using $\nabla f(x)$
- But how many iterations do we need?

Stochastic gradient

Method	Assumptions	Full	Stochastic
Subgradient	convex	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$
Subgradient	strongly cvx	$O(1/k)$	$O(1/k)$

So using stochastic subgradient, solve *n* times faster.

Stochastic gradient

Method	Assumptions	Full	Stochastic
Subgradient	convex	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$
Subgradient	strongly cvx	$O(1/k)$	$O(1/k)$

So using stochastic subgradient, solve *n* times faster.

Method	Assumptions	Full	Stochastic
Gradient	convex	$O(1/k)$	$O(1/\sqrt{k})$
Gradient	strongly cvx	$O((1-\mu/L)^k)$	$O(1/k)$

– For smooth problems, stochastic gradient needs more iterations
– Widely used in ML, rapid initial convergence
– Several speedup techniques studied, but worst case remains same

Hybrid methods

- ▶ Hybrid of stochastic gradient with full gradient.

Stochastic Average Gradient (SAG) (Le Roux, Schmidt, Bach 2012)

- ○ store the gradients of ∇f_i for $i = 1, .., n$
- ○ Select uniformly at random $i(k) \in \{1, \dots, n\}$
- ○ Perform the update

$$x_{k+1} = x_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} y_i^k \quad y_i^k = \begin{cases} \nabla f_i(x_k) & \text{if } i = i(k) \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

Hybrid methods

▶ Hybrid of stochastic gradient with full gradient.

Stochastic Average Gradient (SAG) (Le Roux, Schmidt, Bach 2012)

- store the gradients of ∇f_i for $i = 1, .., n$
- Select uniformly at random $i(k) \in \{1, \ldots, n\}$
- Perform the update

$$x_{k+1} = x_k - \frac{\alpha_k}{n}\sum_{i=1}^{n} y_i^k \quad y_i^k = \begin{cases} \nabla f_i(x_k) & \text{if } i = i(k) \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

- Randomized / stochastic version of incremental gradient method of Blatt et al (2008)
- Storage overhead; acceptable in some ML settings:
  - $f_i(x) = \ell(l_i, x^T\Phi(a_i))$, $\nabla f_i(x) = \nabla\ell(l_i, x^T\Phi(a_i))\Phi(a_i)$
  - Store only n scalars (since depends only on x^Ta_i)

SAG

Method	Assumptions	Rate
Gradient	convex	$O(1/k)$
Gradient	strongly cvx	$O((1-\mu/L)^k)$
Stochastic	strongly cvx	$O(1/k)$
SAG	strongly convex	$O((1-\min\{\frac{\mu}{n},\frac{1}{8n}\})^k)$

This speedup also observed in practice

Complicated convergence analysis

Similar rates for many other methods

– stochastic dual coordinate (SDCA); [Shalev-Shwartz, Zhang, 2013]
– stochastic variance reduced gradient (SVRG); [Johnson, Zhang, 2013]
– proximal SVRG [Xiao, Zhang, 2014]
– hybrid of SAG and SVRG, SAGA (also proximal); [Defazio et al, 2014]
– accelerated versions [Lin, Mairal, Harchoui; 2015]
– incremental Newton method, S2SGD and MS2GD, . . .

SVRG

- For $s \geq 1$:
  1. $\bar{x} \leftarrow \bar{x}_{s-1}$
  2. $\bar{g} \leftarrow \nabla F(\bar{x})$ (full gradient computation)

SVRG

- For $s \geq 1$:

```
1  $\bar{x} \leftarrow \bar{x}_{s-1}$
2  $\bar{g} \leftarrow \nabla F(\bar{x})$                          (full gradient computation)
3  $x_0 = \bar{x}; \quad t \leftarrow \texttt{RAND}(1, m)$          (randomized stopping)
```

SVRG

- For $s \geq 1$:

```
1  $\bar{x} \leftarrow \bar{x}_{s-1}$
2  $\bar{g} \leftarrow \nabla F(\bar{x})$                                  (full gradient computation)
3  $x_0 = \bar{x}; \quad t \leftarrow \mathrm{RAND}(1, m)$                 (randomized stopping)
4  For $k = 0, 1, \ldots, t-1$
       ■ Randomly pick $i(k) \in [1..m]$
       ■ $x_{k+1} = x_k - \eta_k(\nabla f_{i(k)}(x_k) - \nabla f_{i(k)}(\bar{x}) + \bar{g})$
```

SVRG

- For $s \geq 1$:

```
1  $\bar{x} \leftarrow \bar{x}_{s-1}$
2  $\bar{g} \leftarrow \nabla F(\bar{x})$                      (full gradient computation)
3  $x_0 = \bar{x}; \quad t \leftarrow \mathrm{RAND}(1, m)$     (randomized stopping)
4  For $k = 0, 1, \ldots, t-1$
       Randomly pick $i(k) \in [1..m]$
       $x_{k+1} = x_k - \eta_k(\nabla f_{i(k)}(x_k) - \nabla f_{i(k)}(\bar{x}) + \bar{g})$
5  $\bar{x}_s \leftarrow x_t$
```

SVRG

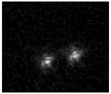
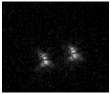
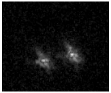
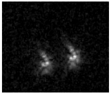
```
For $s \geq 1$:
  1  $\bar{x} \leftarrow \bar{x}_{s-1}$
  2  $\bar{g} \leftarrow \nabla F(\bar{x})$                    (full gradient computation)
  3  $x_0 = \bar{x}; \quad t \leftarrow \mathrm{RAND}(1, m)$      (randomized stopping)
  4  For $k = 0, 1, \ldots, t-1$
       Randomly pick $i(k) \in [1..m]$
       $x_{k+1} = x_k - \eta_k(\nabla f_{i(k)}(x_k) - \nabla f_{i(k)}(\bar{x}) + \bar{g})$
  5  $\bar{x}_s \leftarrow x_t$
```

Theorem Assume each $f_i(x)$ is smooth, and $F(x)$ strongly-convex. Then, for sufficiently large n, there is $\alpha < 1$ s.t.

$$\mathbb{E}[F(\bar{x}_s) - F(x^*)] \leq \alpha^s [F(\bar{x}_0) - F(x^*)]$$

Motivating application

Formulation as matrix factorization

time t	$\boldsymbol{y}_t$	$=$	$\boldsymbol{a}_t$	$*$	$\boldsymbol{x}$	$+$	$\boldsymbol{n}_t$
0		$=$		$*$		$+$	$\boldsymbol{n}_0$
1		$=$		$*$		$+$	$\boldsymbol{n}_1$
2		$=$		$*$		$+$	$\boldsymbol{n}_2$
k		$=$		$*$		$+$	$\boldsymbol{n}_k$

Formulation as matrix factorization

$$\begin{bmatrix} | & \vdots & | \\ y_1 & | & y_n \\ | & \vdots & | \end{bmatrix} \approx \begin{bmatrix} | & \vdots & | \\ a_1 & | & a_t \\ | & \vdots & | \end{bmatrix} * x$$

Rewrite: $a * x = Ax = Xa$

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_t \end{bmatrix} \approx X \begin{bmatrix} a_1 & a_2 & \cdots & a_t \end{bmatrix}$$

$$Y \approx XA$$

Solve this scalably, because...

Scalable matrix factorization

Example, 5000 frames of size 512 × 512

$Y_{262144\times5000} \approx X_{262144\times262144}A_{262144\times5000}$

Without structure $\approx$ 70 billion parameters!

With structure, $\approx$ 4.8 **million parameters!**

Scalable matrix factorization

Example, 5000 frames of size 512 × 512
$Y_{262144 \times 5000} \approx X_{262144 \times 262144} A_{262144 \times 5000}$

Without structure $\approx$ 70 billion parameters!
With structure, $\approx$ 4.8 **million parameters!**

Despite structure, alternating minimization **impractical**
Fix X, solve for A, requires updating $\approx$ 4.5 million params

Scalable matrix factorization

$$\min_{A_t, x} \quad \sum\nolimits_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

Scalable matrix factorization

$$\min_{A_t, x} \quad \sum_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

```
Initialize guess $\boldsymbol{x}_0$
For $t = 1, 2, \ldots$
    1. Observe image $\boldsymbol{y}_t$;
```

Scalable matrix factorization

$$\min_{A_t, x} \quad \sum_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

```
Initialize guess $\boldsymbol{x}_0$
For $t = 1, 2, \ldots$
    1. Observe image $\boldsymbol{y}_t$;
    2. Use $\boldsymbol{x}_{t-1}$ to estimate $\boldsymbol{A}_t$
```

Scalable matrix factorization

$$\min_{A_t, x} \quad \sum_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

```
Initialize guess $\boldsymbol{x}_0$
For $t = 1, 2, \ldots$
    1. Observe image $\boldsymbol{y}_t$;
    2. Use $\boldsymbol{x}_{t-1}$ to **estimate** $\boldsymbol{A}_t$
    3. Solve **optimization subproblem** to obtain $\boldsymbol{x}_t$
```

Scalable matrix factorization

$$\min_{A_t, x} \quad \sum_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

```
Initialize guess $\boldsymbol{x}_0$
For $t = 1, 2, \ldots$
    1. Observe image $\boldsymbol{y}_t$;
    2. Use $\boldsymbol{x}_{t-1}$ to **estimate** $\boldsymbol{A}_t$
    3. Solve **optimization subproblem** to obtain $\boldsymbol{x}_t$
```

Step 2. Model, estimate blur A_t — separate talk

Step 3. convex subproblem — reuse convex building blocks

Do Steps 2, 3 **inexactly** $\Longrightarrow$ realtime processing!

[Harmeling, Hirsch, Sra, Schölkopf (ICCP'09); Hirsch, Sra, Schölkopf, Harmeling (CVPR'10); Hirsch, Harmeling, Sra, Schölkopf (Astron. & Astrophy. (AA) 2011); Harmeling, Hirsch, Sra, Schölkopf, Schuler (Patent 2012); Sra (NIPS'12)]

Algorithmic framework

Key idea

$$\min_{X,A} \Phi(X, A) \quad \equiv \quad \min_{X} \left(\min_{A} \Phi(X, A) \right) =$$

Algorithmic framework

Key idea

$$\begin{aligned}\min_{X,A} \Phi(X,A) &\equiv \min_{X}\left(\min_{A} \Phi(X,A)\right) = \min_{X} F(X)\\ F(X) &:= \min_{A} \Phi(X,A)\end{aligned}$$

Algorithmic framework

Key idea

$$\min_{X,A} \Phi(X,A) \equiv \min_X \left(\min_A \Phi(X,A) \right) = \min_X F(X)$$

$$F(X) := \min_A \Phi(X,A)$$

$$\Phi(X,A) = \|Y - XA\|^2 + \Omega(X) + \Gamma(A)$$

$$\hookrightarrow \min_X \; F(X) + \Omega(X)$$

but now F is **nonconvex**

Inexactness: key to scalability

$$X^{\text{new}} \leftarrow \text{prox}_{\alpha\Omega}(X - \alpha\nabla F(X))$$

Inexactness: key to scalability

$$X^{\text{new}} \leftarrow \operatorname{prox}_{\alpha\Omega}(X - \alpha\nabla F(X) + e\,) + p$$

If gradient is **inexactly** computed

If $\operatorname{prox}_{\Omega}$ **inexactly** computed

Inexactness: key to scalability

$$X^{\text{new}} \leftarrow \operatorname{prox}_{\alpha\Omega}(X - \alpha\nabla F(X) + e\ ) + p$$

If gradient is **inexactly** computed → e

If $\operatorname{prox}_\Omega$ **inexactly** computed → p

Example: Say $F(X) = \sum_{i=1}^m f_i(X)$

Instead of $\nabla F(X)$, use $\nabla f_k(x)$—**incremental!**

m times cheaper (m can be in the millions or more)

Inexactness: key to scalability

incremental prox-method for **large-scale nonconvex**

[Sra (NIPS 12)]; (also *arXiv: [math.OC-1109.0258]*)

Results on real data

Parallel methods

BCD – Setup

$$\boxed{\min f(x) \text{ where } x \in \mathbb{R}^N}$$

BCD – Setup

$$\boxed{\min f(x) \text{ where } x \in \mathbb{R}^N}$$

Assume gradient of block i is Lipschitz continuous

$$\|\nabla_i f(x + E_i h) - \nabla_i f(x)\|_* \leq L_i \|h\|$$

Block gradient $\nabla_i f(x)$ is projection of full grad: $E_i^T \nabla f(x)$

BCD – Setup

$$\boxed{\min f(x) \text{ where } x \in \mathbb{R}^N}$$

Assume gradient of block i is Lipschitz continuous

$$\|\nabla_i f(x + E_i h) - \nabla_i f(x)\|_* \leq L_i \|h\|$$

Block gradient $\nabla_i f(x)$ is projection of full grad: $E_i^T \nabla f(x)$

BCD – Setup

$$\boxed{\min f(x) \text{ where } x \in \mathbb{R}^N}$$

Assume gradient of block i is Lipschitz continuous

$$\|\nabla_i f(x + E_i h) - \nabla_i f(x)\|_* \leq L_i \|h\|$$

Block gradient $\nabla_i f(x)$ is projection of full grad: $E_i^T \nabla f(x)$

Block Coordinate “Gradient” Descent

BCD – Setup

$$\boxed{\min f(x) \text{ where } x \in \mathbb{R}^N}$$

Assume gradient of block i is Lipschitz continuous

$$\|\nabla_i f(x + E_i h) - \nabla_i f(x)\|_* \leq L_i \|h\|$$

Block gradient $\nabla_i f(x)$ is projection of full grad: $E_i^T \nabla f(x)$

Block Coordinate "Gradient" Descent

- Using the descent lemma, we have blockwise upper bounds

$$f(x + E_i h) \leq f(x) + \langle \nabla_i f(x), h\rangle + \tfrac{L_i}{2}\|h\|^2, \quad \text{for } i = 1, \ldots, n.$$

- At each step, minimize these upper bounds!

Randomized BCD

- For $k \geq 0$ (no init. of x necessary)

Randomized BCD

- For $k \geq 0$ (no init. of x necessary)
- Pick a block i from $[n]$ with probability $p_i > 0$

Randomized BCD

- For $k \geq 0$ (no init. of x necessary)
- Pick a block i from $[n]$ with probability $p_i > 0$
- Optimize upper bound (partial gradient step) for block i

$$h = \underset{h}{\operatorname{argmin}}\, f(x_k) + \langle \nabla_i f(x_k),\, h \rangle + \tfrac{L_i}{2}\|h\|^2$$

$$h = -\tfrac{1}{L_i} \nabla_i f(x_k)$$

Randomized BCD

- For $k \geq 0$ (no init. of x necessary)
- Pick a block i from $[n]$ with probability $p_i > 0$
- Optimize upper bound (partial gradient step) for block i

$$
\begin{aligned}
h &= \operatorname*{argmin}_h f(x_k) + \langle \nabla_i f(x_k),\, h \rangle + \tfrac{L_i}{2} \|h\|^2 \\
h &= -\tfrac{1}{L_i} \nabla_i f(x_k)
\end{aligned}
$$

- Update the impacted coordinates of x, formally

Randomized BCD

- For $k \geq 0$ (no init. of x necessary)
- Pick a block i from $[n]$ with probability $p_i > 0$
- Optimize upper bound (partial gradient step) for block i

$$
\begin{aligned}
h &= \operatorname*{argmin}_h f(x_k) + \langle \nabla_i f(x_k), h \rangle + \tfrac{L_i}{2}\|h\|^2 \\
h &= -\tfrac{1}{L_i}\nabla_i f(x_k)
\end{aligned}
$$

- Update the impacted coordinates of x, formally

$$
\begin{aligned}
x_{k+1}^{(i)} &\leftarrow x_k^{(i)} + h \\
x_{k+1} &\leftarrow x_k - \tfrac{1}{L_i} E_i \nabla_i f(x_k)
\end{aligned}
$$

Randomized BCD

- For $k \geq 0$ (no init. of x necessary)
- Pick a block i from $[n]$ with probability $p_i > 0$
- Optimize upper bound (partial gradient step) for block i

$$h = \underset{h}{\operatorname{argmin}}\, f(x_k) + \langle \nabla_i f(x_k),\, h \rangle + \tfrac{L_i}{2}\|h\|^2$$
$$h = -\tfrac{1}{L_i}\nabla_i f(x_k)$$

- Update the impacted coordinates of x, formally

$$x_{k+1}^{(i)} \leftarrow x_k^{(i)} + h$$
$$x_{k+1} \leftarrow x_k - \tfrac{1}{L_i} E_i \nabla_i f(x_k)$$

Notice: Original BCD had: $x_k^{(i)} = \operatorname{argmin}_h f(\ldots, \underbrace{h}_{\text{block } i}, \ldots)$

Randomized BCD

- For $k \geq 0$ (no init. of x necessary)
- Pick a block i from $[n]$ with probability $p_i > 0$
- Optimize upper bound (partial gradient step) for block i

$$h = \underset{h}{\operatorname{argmin}}\, f(x_k) + \langle \nabla_i f(x_k), h \rangle + \tfrac{L_i}{2}\|h\|^2$$

$$h = -\tfrac{1}{L_i}\nabla_i f(x_k)$$

- Update the impacted coordinates of x, formally

$$x_{k+1}^{(i)} \leftarrow x_k^{(i)} + h$$

$$x_{k+1} \leftarrow x_k - \tfrac{1}{L_i} E_i \nabla_i f(x_k)$$

Notice: Original BCD had: $x_k^{(i)} = \operatorname{argmin}_h f(\ldots, \underbrace{h}_{\text{block } i}, \ldots)$

We'll call this BCM (**Block Coordinate Minimization**)

Parallel BCD

Previously

$$\boxed{\min f(x) = f(x_1, \ldots, x_n)}$$

Parallel BCD

Previously

$$\boxed{\min f(x) = f(x_1, \ldots, x_n)}$$

What if?

$$\boxed{\min f(x) = \textstyle\sum_i f_i(x_i)}$$

Parallel BCD

Previously

$$\boxed{\min f(x) = f(x_1, \ldots, x_n)}$$

What if?

$$\boxed{\min f(x) = \sum_i f_i(x_i)}$$

- Can solve all n problems **independently** in **parallel**
- In theory: n times speedup possible compared to serial case

Parallel BCD

Previously

$$\boxed{\min f(x) = f(x_1, \ldots, x_n)}$$

What if?

$$\boxed{\min f(x) = \sum_i f_i(x_i)}$$

- Can solve all n problems **independently** in **parallel**
- In theory: n times speedup possible compared to serial case
- So if objective functions are “almost separable” we would still expect high speedup, diminished by amount of **separability**
- Big data problems often have this “almost separable” structure!

Partial Separability

Consider the **sparse** data matrix

$$\begin{pmatrix} d_{11} & d_{12} & & & \\ & d_{22} & d_{23} & & \\ & & \ddots & \ddots & \end{pmatrix} \in \mathbb{R}^{m\times n},$$

Partial Separability

Consider the **sparse** data matrix

$$\begin{pmatrix} d_{11} & d_{12} & & & \\ & d_{22} & d_{23} & & \\ & & \ddots & \ddots & \end{pmatrix} \in \mathbb{R}^{m\times n},$$

- Objective $f(x) = \|Dx - b\|_2^2 = \sum_{i=1}^m (d_i^T x - b_i)^2$ also equals
$$(d_{11}x_1 + d_{12}x_2 - b_1)^2 + (d_{22}x_2 + d_{23}x_3 - b_2)^2 + \cdots$$
- Each term depends on only 2 coordinates
- Formally, we could write this as
$$f(x) = \sum\nolimits_{J\in\mathcal{J}} f_J(x),$$
where $\mathcal{J} = \{\{1,2\},\{2,3\},\cdots\}$
- Key point: $f_J(x)$ depends only on x_j for $j \in J$.

Partial Separability

$$\boxed{\min f(x) \text{ s.t. } x \in \mathbb{R}^n}$$

Def. Let $\mathcal{J}$ be a collection of subsets of $\{1, \ldots, n\}$. We say f is **partially separable of degree** ω if it can be written as

$$f(x) = \sum_{J \in \mathcal{J}} f_J(x),$$

where each f_J depends only on x_j for $j \in J$, and

$$|J| \le \omega \quad \forall J \in \mathcal{J}.$$

Example: If $D_{m \times n}$ is a sparse matrix, then $\omega = \max_{1 \le i \le m} \|d_i^T\|_0$

Partial Separability

$$\boxed{\min f(x) \text{ s.t. } x \in \mathbb{R}^n}$$

Def. Let $\mathcal{J}$ be a collection of subsets of $\{1, \ldots, n\}$. We say f is **partially separable of degree** ω if it can be written as

$$f(x) = \sum_{J \in \mathcal{J}} f_J(x),$$

where each f_J depends only on x_j for $j \in J$, and

$$|J| \leq \omega \quad \forall J \in \mathcal{J}.$$

Example: If $D_{m\times n}$ is a sparse matrix, then $\omega = \max_{1 \leq i \leq m} \|d_i^T\|_0$

Exercise: Extend this notion to $x = (x^{(1)}, \ldots, x^{(n)})$

Hint: Now, f_J will depend only on $x^{(j)}$ for $j \in J$

Parallel Stochastic Gradient!

Each core runs the computation:

1. Sample coordinates J from $\{1, \ldots, n\}$ (all sets of variables)
2. Read current state of x_J from shared memory
3. For each individual coordinate $j \in J$
   $x_j \leftarrow x_j - \alpha_k[\nabla f_J(x_J)]_j$

Parallel Stochastic Gradient!

Each core runs the computation:

1. Sample coordinates J from $\{1, \ldots, n\}$ (all sets of variables)
2. Read current state of x_J from shared memory
3. For each individual coordinate $j \in J$
   $x_j \leftarrow x_j - \alpha_k[\nabla f_J(x_J)]_j$

- **Atomic update** only for $x_j \leftarrow x_j - a$ (not for gradient)

Parallel Stochastic Gradient!

Each core runs the computation:

1. Sample coordinates J from $\{1, \ldots, n\}$ (all sets of variables)
2. Read current state of x_J from shared memory
3. For each individual coordinate $j \in J$
   $x_j \leftarrow x_j - \alpha_k [\nabla f_J(x_J)]_j$

- **Atomic update** only for $x_j \leftarrow x_j - a$ (not for gradient)
- Since the actual coordinate j can arise in various J, processors can overwrite each others' work.

Parallel Stochastic Gradient!

Each core runs the computation:

1. Sample coordinates J from $\{1, \ldots, n\}$ (all sets of variables)
2. Read current state of x_J from shared memory
3. For each individual coordinate $j \in J$
   $x_j \leftarrow x_j - \alpha_k[\nabla f_J(x_J)]_j$

- **Atomic update** only for $x_j \leftarrow x_j - a$ (not for gradient)
- Since the actual coordinate j can arise in various J, processors can overwrite each others' work.
- But if **partial overlaps (separability)**, coordinate j does not appear in too many different subsets J, method works fine!

Parallel BCD

1. Choose initial point $x_0 \in \mathbb{R}^N$

Parallel BCD

1. Choose initial point $x_0 \in \mathbb{R}^N$
2. For $k \geq 0$
   - Randomly pick (in parallel) a set of blocks $S_k \subset \{1, \ldots, n\}$

Parallel BCD

1. Choose initial point $x_0 \in \mathbb{R}^N$
2. For $k \geq 0$
   - Randomly pick (in parallel) a set of blocks $S_k \subset \{1, \ldots, n\}$
   - Perform BCD updates (in parallel) for $i \in S_k$

$$x_{k+1}^{(i)} \leftarrow x_k^{(i)} - \frac{1}{\beta w_i} \nabla_i f(x_k)$$

$\longrightarrow$ w_i typically L_i; β depends on degree of separability ω

Parallel BCD

1. Choose initial point $x_0 \in \mathbb{R}^N$
2. For $k \geq 0$
   - Randomly pick (in parallel) a set of blocks $S_k \subset \{1, \ldots, n\}$
   - Perform BCD updates (in parallel) for $i \in S_k$

$$x_{k+1}^{(i)} \leftarrow x_k^{(i)} - \frac{1}{\beta w_i} \nabla_i f(x_k)$$

$\longrightarrow$ w_i typically L_i; β depends on degree of separability ω

♠ Uniform sampling of blocks (or just coordinates)

♠ More careful sampling leads to better guarantees

Parallel BCD

1. Choose initial point $x_0 \in \mathbb{R}^N$
2. For $k \geq 0$
   - Randomly pick (in parallel) a set of blocks $S_k \subset \{1, \ldots, n\}$
   - Perform BCD updates (in parallel) for $i \in S_k$

$$x_{k+1}^{(i)} \leftarrow x_k^{(i)} - \frac{1}{\beta w_i} \nabla_i f(x_k)$$

$\longrightarrow$ w_i typically L_i; β depends on degree of separability ω

♠ Uniform sampling of blocks (or just coordinates)

♠ More careful sampling leads to better guarantees

♠ Theory requires **atomic updates**

Parallel BCD

1. Choose initial point $x_0 \in \mathbb{R}^N$
2. For $k \geq 0$
   - Randomly pick (in parallel) a set of blocks $S_k \subset \{1, \ldots, n\}$
   - Perform BCD updates (in parallel) for $i \in S_k$

$$x_{k+1}^{(i)} \leftarrow x_k^{(i)} - \frac{1}{\beta w_i} \nabla_i f(x_k)$$

$\longrightarrow$ w_i typically L_i; β depends on degree of separability ω

♠ Uniform sampling of blocks (or just coordinates)

♠ More careful sampling leads to better guarantees

♠ Theory requires **atomic updates**

♠ Useful to implement **asynchronously** (i.e., use whatever latest $x^{(i)}$ a given core has access to)

♠ Theory of above method requires **guaranteed descent**

♠ Newer asynchronous CD methods also exist (see survey by Wright, 2015)

Parallel computation – high level views

- Intuition from above: degree of separability strongly correlated with degree of parallelism possible

Parallel computation – high level views

- Intuition from above: degree of separability strongly correlated with degree of parallelism possible
- Not insisting on exact computation allows more parallelism

Parallel computation – high level views

- Intuition from above: degree of separability strongly correlated with degree of parallelism possible
- Not insisting on exact computation allows more parallelism
- Suppose f is the fraction of sequential computation. Then speedup for **any** number of processors (cores) is $\leq 1/f$

Parallel computation – high level views

- Intuition from above: degree of separability strongly correlated with degree of parallelism possible
- Not insisting on exact computation allows more parallelism
- Suppose f is the fraction of sequential computation. Then speedup for **any** number of processors (cores) is $\leq 1/f$
- Parallel optimization on multi-core machines: shared memory architecture. Main penalty: synchronization / atomic operations

Parallel computation – high level views

- Intuition from above: degree of separability strongly correlated with degree of parallelism possible
- Not insisting on exact computation allows more parallelism
- Suppose f is the fraction of sequential computation. Then speedup for **any** number of processors (cores) is $\leq 1/f$
- Parallel optimization on multi-core machines: shared memory architecture. Main penalty: synchronization / atomic operations
- Distributed optimization across machines: synchronization and communication biggest burden;

Parallel computation – high level views

- Intuition from above: degree of separability strongly correlated with degree of parallelism possible
- Not insisting on exact computation allows more parallelism
- Suppose f is the fraction of sequential computation. Then speedup for **any** number of processors (cores) is $\leq 1/f$
- Parallel optimization on multi-core machines: shared memory architecture. Main penalty: synchronization / atomic operations
- Distributed optimization across machines: synchronization and communication biggest burden; node failure, network failure, load-balancing, etc.

Parallel computation – high level views

- Intuition from above: degree of separability strongly correlated with degree of parallelism possible
- Not insisting on exact computation allows more parallelism
- Suppose f is the fraction of sequential computation. Then speedup for **any** number of processors (cores) is $\leq 1/f$
- Parallel optimization on multi-core machines: shared memory architecture. Main penalty: synchronization / atomic operations
- Distributed optimization across machines: synchronization and communication biggest burden; node failure, network failure, load-balancing, etc.
- Synchronous vs. asynchronous computation

Poor man's parallelism

Separable optimization

$$\min \quad f(x) := \sum_{i=1}^{m} f_i(x) \quad x \in \mathbb{R}^n.$$

Separable optimization

$$\min \quad f(x) := \sum_{i=1}^{m} f_i(x) \quad x \in \mathbb{R}^n.$$

Product space trick

Separable optimization

$$\min \quad f(x) := \sum_{i=1}^{m} f_i(x) \quad x \in \mathbb{R}^n.$$

Product space trick

- Introduce (local) variables $(x_1, \ldots, x_m)$

Separable optimization

$$\min \quad f(x) := \sum_{i=1}^{m} f_i(x) \quad x \in \mathbb{R}^n.$$

Product space trick

- Introduce (local) variables $(x_1, \ldots, x_m)$
- Problem is now over $\mathcal{H}^m := \mathcal{H} \times \mathcal{H} \times \cdots \times \mathcal{H}$ (m-times)

Separable optimization

$$\min \quad f(x) := \sum_{i=1}^{m} f_i(x) \quad x \in \mathbb{R}^n.$$

Product space trick

- Introduce (local) variables $(x_1, \ldots, x_m)$
- Problem is now over $\mathcal{H}^m := \mathcal{H} \times \mathcal{H} \times \cdots \times \mathcal{H}$ (m-times)
- **Consensus** constraint: $x_1 = x_2 = \ldots = x_m$

$$\begin{aligned} \min_{(x_1,\ldots,x_m)} \quad & \sum\nolimits_i f_i(x_i) \\ \text{s.t.} \quad & x_1 = x_2 = \cdots = x_m. \end{aligned}$$

Separable optimization

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \mathbb{1}_{\mathcal{B}}(\boldsymbol{x})$$

where $\boldsymbol{x} \in \mathcal{H}^m$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathcal{H}^m \mid \boldsymbol{z} = (x, x, \ldots, x)\}$

Separable optimization

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \mathbb{1}_{\mathcal{B}}(\boldsymbol{x})$$

where $\boldsymbol{x} \in \mathcal{H}^m$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathcal{H}^m \mid \boldsymbol{z} = (x, x, \ldots, x)\}$

- Can solve using proximal splitting methods (e.g., DR, ADMM)

Separable optimization

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \mathbb{1}_{\mathcal{B}}(\boldsymbol{x})$$

where $\boldsymbol{x} \in \mathcal{H}^m$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathcal{H}^m \mid \boldsymbol{z} = (x, x, \ldots, x)\}$

- Can solve using proximal splitting methods (e.g., DR, ADMM)
- Each component of $f_i(x_i)$ independently in parallel
- Communicate / synchronize to ensure consensus
- Asynchronous versions exist (results from 2014, 2015)