Subgradient methods for huge-scale optimization problems

Yurii Nesterov, CORE/INMA (UCL)

May 24, 2012 (Edinburgh, Scotland)

Outline

Nonlinear Optimization: problems sizes

Class	Operations	Dimension	Iter.Cost	Memory	
Small-size	All	$10^0 - 10^2$	$n^4 \to n^3$	Kilobyte:	10^3
Medium-size	A^{-1}	$10^3 - 10^4$	$n^3 \to n^2$	Megabyte:	10^6
Large-scale	Ax	$10^5 - 10^7$	$n^2 \to n$	Gigabyte:	10^9
Huge-scale	$x + y$	$10^8 - 10^{12}$	$n \to \log n$	Terabyte:	10^{12}

Sources of Huge-Scale problems

- Internet (New)
- Telecommunications (New)
- Finite-element schemes (Old)
- PDE, Weather prediction (Old)

Main hope: Sparsity.

Sparse problems

Problem: $\min\limits_{x \in Q} f(x)$, where Q is closed and convex in R^N, and

- $f(x) = \Psi(Ax)$, where Ψ is a simple *convex function*:

$$\Psi(y_1) \geq \Psi(y_2) + \langle \Psi'(y_2), y_1 - y_2 \rangle, \quad y_1, y_2 \in R^M,$$

- $A : R^N \to R^M$ is a *sparse* matrix.

Let $p(x) \overset{\text{def}}{=} \#$ of nonzeros in x. <u>Sparsity coefficient:</u> $\gamma(A) \overset{\text{def}}{=} \frac{p(A)}{MN}$.

Example 1: Matrix-vector multiplication

- Computation of vector Ax needs $p(A)$ operations.
- Initial complexity MN is reduced in $\gamma(A)$ times.

Example: Gradient Method

$$x_0 \in Q, \quad x_{k+1} = \pi_Q(x_k - hf'(x_k)), \quad k \geq 0.$$

Main computational expenses

- Projection of simple set Q needs $O(N)$ operations.
- Displacement $x_k \to x_k - hf'(x_k)$ needs $O(N)$ operations.
- $f'(x) = A^T \Psi'(Ax)$. If Ψ is simple, then the main efforts are spent for two matrix-vector multiplications: $2p(A)$.

Conclusion: As compared with *full* matrices, we accelerate in $\gamma(A)$ times.
Note: For Large- and Huge-scale problems, we often have
$\gamma(A) \approx 10^{-4} \dots 10^{-6}$. **Can we get more?**

Sparse updating strategy

Main idea

- After update $x_+ = x + d$ we have $y_+ \overset{\text{def}}{=} Ax_+ = \underbrace{Ax}_{y} + Ad$.
- What happens if d is *sparse*?

Denote $\sigma(d) = \{j : d^{(j)} \neq 0\}$. Then $y_+ = y + \sum\limits_{j \in \sigma(d)} d^{(j)} \cdot Ae_j$.

Its complexity, $\kappa_A(d) \overset{\text{def}}{=} \sum\limits_{j \in \sigma(d)} p(Ae_j)$, can be VERY small!

$$\begin{aligned} \kappa_A(d) &= M \sum_{j \in \sigma(d)} \gamma(Ae_j) = \gamma(d) \cdot \frac{1}{p(d)} \sum_{j \in \sigma(d)} \gamma(Ae_j) \cdot MN \\ &\leq \gamma(d) \max_{1 \leq j \leq m} \gamma(Ae_j) \cdot MN. \end{aligned}$$

If $\gamma(d) \leq c\gamma(A)$, $\gamma(A_j) \leq c\gamma(A)$, then

$\boxed{\kappa_A(d) \leq c^2 \cdot \gamma^2(A) \cdot MN}$.

Expected acceleration: $(10^{-6})^2 = 10^{-12} \Rightarrow 1$ sec $\approx 32\,000$ years!

When it can work?

- Simple methods: No full-vector operations! (Is it possible?)
- Simple problems: Functions with *sparse* gradients.

Let us try:

1. Quadratic function $f(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$. The gradient
$$f'(x) = Ax - b, \quad x \in R^N,$$
is *not* sparse even if A is sparse.
2. Piece-wise linear function $g(x) = \max\limits_{1 \le i \le m} [\langle a_i, x\rangle - b^{(i)}]$. Its *subgradient* $f'(x) = a_{i(x)}$, $i(x)$: $f(x) = \langle a_{i(x)}, x\rangle - b^{(i(x))}$, can be sparse is a_i is sparse!

But: We need a fast procedure for updating *max-type operations*.

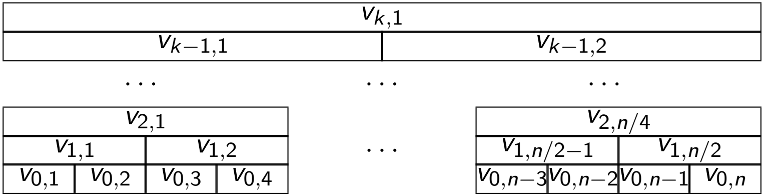
Fast updates in short computational trees

Def: Function $f(x)$, $x \in R^n$, is *short-tree representable*, if it can be computed by a short binary tree with the height $\approx \ln n$.

Let $n = 2^k$ and the tree has $k+1$ levels: $v_{0,i} = x^{(i)}$, $i = 1, \dots, n$. Size of the next level halves the size of the previous one:

$$v_{i+1,j} \;=\; \psi_{i+1,j}(v_{i,2j-1}, v_{i,2j}), \quad j = 1, \dots, 2^{k-i-1}, \; i = 0, \dots, k-1,$$

where $\psi_{i,j}$ are some bivariate functions.



Main advantages

- Important examples (symmetric functions)

$$\begin{aligned}
f(x) &= \|x\|_p, \quad p \geq 1, & \psi_{i,j}(t_1,t_2) &\equiv [\, |t_1|^p + |t_2|^p \,]^{1/p}, \\
f(x) &= \ln\left(\sum_{i=1}^n e^{x^{(i)}}\right), & \psi_{i,j}(t_1,t_2) &\equiv \ln\left(e^{t_1} + e^{t_2}\right), \\
f(x) &= \max_{1\leq i\leq n} x^{(i)}, & \psi_{i,j}(t_1,t_2) &\equiv \max\{t_1,t_2\}.
\end{aligned}$$

- The binary tree requires only $n-1$ auxiliary cells.
- Its value needs $n-1$ applications of $\psi_{i,j}(\cdot,\cdot)$ ($\equiv$ *operations*).
- If x_+ differs from x in one entry only, then for re-computing $f(x_+)$ we need only $k \equiv \log_2 n$ operations.

Thus, we can have pure subgradient minimization schemes with

Sublinear Iteration Cost

.

Simple subgradient methods

I. Problem: $f^* \stackrel{\mathrm{def}}{=} \min\limits_{x \in Q} f(x)$, where

- Q is a closed and convex and $\|f'(x)\| \le L(f)$, $x \in Q$,
- the optimal value f^* is known.

Consider the following optimization scheme (B.Polyak, 1967):

$$x_0 \in Q, \quad x_{k+1} = \pi_Q \left(x_k - \frac{f(x_k) - f^*}{\|f'(x_k)\|^2} f'(x_k) \right), \quad k \ge 0.$$

Denote $f_k^* = \min\limits_{0 \le i \le k} f(x_i)$. Then for any $k \ge 0$ we have:

$$f_k^* - f^* \le \frac{L(f)\|x_0 - \pi_{X_*}(x_0)\|}{(k+1)^{1/2}},$$

$$\|x_k - x^*\| \le \|x_0 - x^*\|, \quad \forall x^* \in X_*.$$

Proof:

Let us fix $x^* \in X_*$. Denote $r_k(x^*) = \|x_k - x^*\|$. Then

$$\begin{array}{rcl} r_{k+1}^2(x^*) & \leq & \left\| x_k - \frac{f(x_k)-f^*}{\|f'(x_k)\|^2} f'(x_k) - x^* \right\|^2 \\ & = & r_k^2(x^*) - 2\frac{f(x_k)-f^*}{\|f'(x_k)\|^2} \langle f'(x_k), x_k - x^* \rangle + \frac{(f(x_k)-f^*)^2}{\|f'(x_k)\|^2} \\ & \leq & r_k^2(x^*) - \frac{(f(x_k)-f^*)^2}{\|f'(x_k)\|^2} \;\leq\; r_k^2(x^*) - \frac{(f_k^*-f^*)^2}{L^2(f)}. \end{array}$$

From this reasoning, $\|x_{k+1} - x^*\|^2 \leq \|x_k - x^*\|^2$, $\forall x^* \in X^*$. □

Corollary: Assume X_* has recession direction d_*. Then

$$\|x_k - \pi_{X_*}(x_0)\| \leq \|x_0 - \pi_{X_*}(x_0)\|, \quad \langle d_*, x_k \rangle \geq \langle d_*, x_0 \rangle.$$

(Proof: consider $x^* = \pi_{X_*}(x_0) + \alpha d_*$, $\alpha \geq 0$.) □

Constrained minimization (N.Shor (1964) & B.Polyak)

II. Problem: $\min\limits_{x \in Q} \{ f(x) : \; g(x) \leq 0 \}$, where

- Q is closed and convex,
- f, g have uniformly bounded subgradients.

Consider the following method. It has step-size parameter $h > 0$.

If $g(x_k) > h \| g'(x_k) \|$, then (A): $x_{k+1} = \pi_Q \left(x_k - \frac{g(x_k)}{\| g'(x_k) \|^2} \, g'(x_k) \right)$,

else (B): $x_{k+1} = \pi_Q \left(x_k - \frac{h}{\| f'(x_k) \|} \, f'(x_k) \right)$.

Let $\mathcal{F}_k \subseteq \{0, \dots, k\}$ be the set (B)-iterations, and $f_k^* = \min\limits_{i \in \mathcal{F}_k} f(x_i)$.

Theorem: If $k > \| x_0 - x^* \|^2 / h^2$, then $\mathcal{F}_k \neq \emptyset$ and

$$f_k^* - f(x) \leq hL(f), \quad \max_{i \in \mathcal{F}_k} g(x_i) \;\leq\; hL(g).$$

Computational strategies

1. Constants $L(f)$, $L(g)$ are known (e.g. Linear Programming)

We can take $h = \frac{\epsilon}{\max\{L(f), L(g)\}}$. Then we need to decide on the number of steps N (easy!).

Note: The standard advice is $h = \frac{R}{\sqrt{N+1}}$ (much more difficult!)

2. Constants $L(f)$, $L(g)$ are not known

- Start from a guess.
- Restart from scratch each time we see the guess is wrong.
- The guess is doubled after restart.

3. Tracking the record value f_k^*

Double run. Other ideas are welcome!

Random sparse coordinate methods

III. Problem: $\min\limits_{x \geq 0} \left\{ f(x) \overset{\mathrm{def}}{=} \max\limits_{1 \leq i \leq M} [\ell_i(x) \equiv \langle a_i, x \rangle - b_i] \right\}.$

Define $i(x)$: $f(x) = \ell_{i(x)}(x)$, and random variable $\xi(x)$, which gives indexes from $\sigma(a_{i(x)})$ with equal probabilities $\frac{1}{p(a_{i(x)})}$.

Assuming that f^* is known, we can define now a random vector variable $\mathrm{Next}(x)$ by the following rules:

1. Compute $h(x) = \frac{f(x) - f^*}{\|f'(x)\|^2}$. Generate $j(x) = \xi(x)$.
2. Define $[\ \mathrm{Next}(x)\ ]^{(j(x))} = \left(x^{(j(x))} - h(x) a_{i(x)}^{(j(x))} \right)_+$.
3. For other indexes $j \neq j(x)$, define $[\ \mathrm{Next}(x)\ ]^{(j)} = x^{(j)}$.

Algorithmic scheme

0. Choose $x_0 \geq 0$. Compute $u_0 = Ax_0 - b$ and $f(x_0)$.

1. ***k*th iteration ($k \geq 0$).**

a) Generate $j_k = \xi(x_k)$ and update $x_{k+1} = \mathrm{Next}(x_k)$.

b) Update $u_{k+1} = u_k + Ae_{j_k} \cdot \left(x_{k+1}^{(j_k)} - x_k^{(j_k)} \right)$, re-computing in parallel the value of $f(x_{k+1})$.

This method defines a sequence of discrete random variables $\{x_k\}$. Denote $f_k^* = \min\limits_{0 \leq i \leq k} f(x_i)$.

Theorem: Let $p(a_i) \leq r$, $i = 1, \dots, m$. Then, for any $k \geq 0$ we have:

$$\begin{array}{rcl} \mathcal{E}\left([f_k^* - f^*]^2 \right) & \leq & \frac{rL^2(f) \|x_0 - \pi_{X_*}(x_0)\|^2}{k+1}, \\ \mathcal{E}(\|x_k - x_*\|^2) & \leq & \|x_0 - x_*\|^2, \quad \forall x_* \in X_*. \end{array}$$

NB: One iteration needs at most $\max\limits_{1 \leq j \leq N} p(Ae_j) \cdot \log_2 M$ operations.

Application examples

Observations:

1. Very often, Large- and Huge- scale problems have repetitive sparsity patterns and/or limited connectivity.
   - Social networks.
   - Mobile phone networks.
   - Truss topology design (local bars).
   - Finite elements models (2D: four neighbors, 3D: six neighbors).
2. For p-diagonal matrices $\kappa(A) \leq p^2$.

Google problem

Goal: Rank the agents in the society by their social weights.

- Unknown: $x_i \geq 0$ - social influence of agent $i = 1, \ldots, N$.
- Known: σ_i - set of friends of agent i.

Hypothesis

- Agent i shares his support among all friends by equal parts.
- The influence of agent i is equal to the total support obtained from his friends.

Mathematical formulation: quadratic problem

Let $E \in R^{N \times N}$ be an incidence matrix of the connections graph.
Denote $e = (1, \dots, 1)^T \in R^N$ and $\bar{E} = E \cdot \mathrm{diag}\,(E^T e)^{-1}$.
Since, $\bar{E}^T e = e$, this matrix is stochastic.

Problem: Find $x^* \geq 0 : \quad \bar{E} x^* = x^*, \quad x^* \neq 0.$
The size is very big!

Known technique:

- Regularization + Fixed Point (Google Founders, B.Polyak & coauthors, etc.)
- **N09:** Solve it by random CD-method as applied to $\frac{1}{2}\|\bar{E}x - x\|^2 + \frac{\gamma}{2}[\langle e, x\rangle - 1]^2, \quad \gamma > 0.$

  Main drawback: No interpretation for the objective function!

Nonsmooth formulation of Google Problem

Main property of spectral radius ($A \geq 0$)

If $A \in R_+^{n \times n}$, then $\rho(A) = \min\limits_{x \geq 0} \max\limits_{1 \leq i \leq n} \frac{1}{x^{(i)}} \langle e_i, Ax \rangle$.
The minimum is attained at the corresponding eigenvector.

Since $\rho(\bar{E}) = 1$, our problem is as follows:

$$f(x) \overset{\text{def}}{=} \max_{1 \leq i \leq N} [\langle e_i, \bar{E}x \rangle - x^{(i)}] \quad \to \quad \min_{x \geq 0} .$$

Interpretation: Increase self-confidence!
Since $f^* = 0$, we can apply Polyak's method with sparse updates.
Additional features; the optimal set X^* is a *convex cone*.
If $x_0 = e$, then the whole sequence is separated from zero:

$$\langle x^*, e \rangle \leq \langle x^*, x_k \rangle \leq \|x^*\|_1 \cdot \|x_k\|_\infty = \langle x^*, e \rangle \cdot \|x_k\|_\infty .$$

Goal: Find $\bar{x} \geq 0$ such that $\|\bar{x}\|_\infty \geq 1$ and $f(\bar{x}) \leq \epsilon$.
(First condition is satisfied automatically.)

Computational experiments: Iteration Cost

We compare Polyak's GM with sparse update (GM_s) with the standard one (GM).

Setup: Each agent has exactly p random friends.

Thus, $\kappa(A) \stackrel{\mathrm{def}}{=} \max\limits_{1 \le i \le M} \kappa_A(A^T e_i) \approx p^2$.

Iteration Cost: $GM_s \le \kappa(A) \log_2 N \approx p^2 \log_2 N, \quad GM \approx pN.$

$(\log_2 \ 10^3 = 10, \quad \log_2 \ 10^6 = 20, \quad \log_2 \ 10^9 = 30)$

Time for 10^4 iterations ($p = 32$)

N	$\kappa(A)$	GM_s	GM
1024	1632	3.00	2.98
2048	1792	3.36	6.41
4096	1888	3.75	15.11
8192	1920	4.20	139.92
16384	1824	4.69	408.38

Time for 10^3 iterations ($p = 16$)

N	$\kappa(A)$	GM_s	GM
131072	576	0.19	213.9
262144	592	0.25	477.8
524288	592	0.32	1095.5
1048576	608	0.40	2590.8

1 sec $\approx$ 100 min!

Convergence of GM_s: Medium Size

Let $N = 131072$, $p = 16$, $\kappa(A) = 576$, and $L(f) = 0.21$.

Iterations	$f - f^*$	Time (sec)
$1.0 \cdot 10^5$	0.1100	16.44
$3.0 \cdot 10^5$	0.0429	49.32
$6.0 \cdot 10^5$	0.0221	98.65
$1.1 \cdot 10^6$	0.0119	180.85
$2.2 \cdot 10^6$	0.0057	361.71
$4.1 \cdot 10^6$	0.0028	674.09
$7.6 \cdot 10^6$	0.0014	1249.54
$1.0 \cdot 10^7$	0.0010	1644.13

Dimension and accuracy are sufficiently high, but the time is still reasonable.

Convergence of GM_s: Large Scale

Let $N = 1048576$, $p = 8$, $\kappa(A) = 192$, and $L(f) = 0.21$.

Iterations	$f - f^*$	Time (sec)
0	2.000000	0.00
$1.0 \cdot 10^5$	0.546662	7.69
$4.0 \cdot 10^5$	0.276866	30.74
$1.0 \cdot 10^6$	0.137822	76.86
$2.5 \cdot 10^6$	0.063099	192.14
$5.1 \cdot 10^6$	0.032092	391.97
$9.9 \cdot 10^6$	0.016162	760.88
$1.5 \cdot 10^7$	0.010009	1183.59

Final point $\bar{x}_*$: $\|\bar{x}_*\|_\infty = 2.941497$, $R_0^2 \overset{\text{def}}{=} \|\bar{x}_* - e\|_2^2 = 1.2 \cdot 10^5$.

Theoretical bound: $\frac{L^2(f) R_0^2}{\epsilon^2} = 5.3 \cdot 10^7$. **Time for GM:** $\approx$ 1 year!

Conclusion

1. Sparse GM is an efficient and reliable method for solving Large- and Huge- Scale problems with uniform sparsity.
2. We can treat also dense rows. Assume that inequality $\langle a, x \rangle \leq b$ is dense. It is equivalent to the following *system*:

$$\begin{array}{rcl} y^{(1)} & = & a^{(1)} x^{(1)}, \quad y^{(j)} = y^{(j-1)} + a^{(j)} x^{(j)}, \quad j = 2, \dots, n, \\ y^{(n)} & \leq & b. \end{array}$$

   We need *new variables* $y^{(j)}$ for all nonzero coefficients of a.

   - Introduce $p(a)$ additional variables and $p(A)$ additional equality constraints. (No problem!)
   - Hidden drawback: the above equalities are satisfied with *errors*.
   - May be it is not too bad?
3. Similar technique can be applied to dense columns.

Theoretical consequences

Assume that $\kappa(A) \approx \gamma^2(A) n^2$. Compare three methods:

- Sparse updates (SU). Complexity $\gamma^2(A) n^2 \frac{L^2 R^2}{\epsilon^2} \log n$ operations.
- Smoothing technique (ST). Complexity $\gamma(A) n^2 \frac{LR}{\epsilon}$ operations.
- Polynomial-time methods (PT). Complexity $(\gamma(A) n + n^3) n \ln \frac{LR}{\epsilon}$ operations.

There are three possibilities.

- Low accuracy: $\gamma(A) \frac{LR}{\epsilon} < 1$. Then we choose SU.
- Moderate accuracy: $1 < \gamma(A) \frac{LR}{\epsilon} < n^2$. We choose ST.
- High accuracy: $\gamma(A) \frac{LR}{\epsilon} > n^2$. We choose PT.

NB: For Huge-Scale problems usually $\gamma(A) \approx \frac{1}{n}$.