Lecture 3: Huge-scale optimization problems

Yurii Nesterov, CORE/INMA (UCL)

September 10, 2012

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)
- Partial differential equations (Old)

Very old optimization idea: Coordinate Search

Problem: $\min\limits_{x \in R^n} f(x)$ (f is convex and differentiable).

Coordinate relaxation algorithm

For $k \geq 0$ iterate

1. Choose active coordinate i_k.
2. Update $x_{k+1} = x_k - h_k \nabla_{i_k} f(x_k) e_{i_k}$ ensuring $f(x_{k+1}) \leq f(x_k)$. (e_i is ith coordinate vector in R^n.)

Main advantage: Very simple implementation.

Possible strategies

1. Cyclic moves. (Difficult to analyze.)
2. Random choice of coordinate (Why?)
3. Choose coordinate with the maximal directional derivative.

Complexity estimate: assume

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad x, y \in R^n.$$

Let us choose $h_k = \frac{1}{L}$. Then

$$\begin{aligned} f(x_k) - f(x_{k+1}) \ &\geq \ \tfrac{1}{2L}|\nabla_{i_k} f(x_k)|^2 \ \geq \ \tfrac{1}{2nL}\|\nabla f(x_k)\|^2 \\ &\geq \ \tfrac{1}{2nLR^2}(f(x_k) - f^*)^2. \end{aligned}$$

Hence, $f(x_k) - f^* \leq \frac{2nLR^2}{k}$, $k \geq 1$. (For Grad.Method, drop n.)

This is the only known theoretical result known for CDM!

Criticism

Theoretical justification:

- Complexity bounds are not known for the most of the schemes.
- The only justified scheme needs computation of the <u>whole gradient</u>. (Why don't use GM?)

Computational complexity:

- <u>Fast differentiation:</u> if function is defined by a sequence of operations, then $C(\nabla f) \leq 4C(f)$.
- Can we do anything without computing the function's values?

Result: CDM are almost out of the computational practice.

Google problem

Let $E \in R^{n\times n}$ be an incidence matrix of a graph. Denote $e = (1,\ldots,1)^T$ and

$$\bar{E} = E \cdot \mathrm{diag}\,(E^T e)^{-1}.$$

Thus, $\bar{E}^T e = e$. Our problem is as follows:

$$\text{Find } x^* \geq 0 : \quad \bar{E}x^* = x^*.$$

Optimization formulation:

$$f(x) \overset{\mathrm{def}}{=} \tfrac{1}{2}\|\bar{E}x - x\|^2 + \tfrac{\gamma}{2}[\langle e, x\rangle - 1]^2 \quad \rightarrow \quad \min_{x\in R^n}$$

Huge-scale problems

Main features

- The size is very big ($n \geq 10^7$).
- The data is distributed in space.
- The requested parts of data are not always available.
- The data may be changing in time.

Consequences

Simplest operations are expensive or infeasible:

- Update of the full vector of variables.
- Matrix-vector multiplication.
- Computation of the objective function's value, etc.

Structure of the Google Problem

Let ua look at the gradient of the objective:

$$\begin{aligned} \nabla_i f(x) &= \langle a_i, g(x)\rangle + \gamma[\langle e, x\rangle - 1], \; i = 1, \dots, n, \\ g(x) &= \bar{E}x - x \in R^n, \quad (\bar{E} = (a_1, \dots, a_n)). \end{aligned}$$

Main observations:

- The coordinate move $x_+ = x - h_i \nabla_i f(x) e_i$ needs $O(p_i)$ a.o. (p_i is the number of nonzero elements in a_i.)
- $d_i \overset{\mathrm{def}}{=} \mathrm{diag}\left(\nabla^2 f \overset{\mathrm{def}}{=} \bar{E}^T\bar{E} + \gamma e e^T\right)_i = \gamma + \frac{1}{p_i}$ are available. We can use them for choosing the step sizes ($h_i = \frac{1}{d_i}$).

Reasonable coordinate choice strategy? Random!

Random coordinate descent methods (RCDM)

$$\min_{x \in R^N} f(x), \quad (f \text{ is convex and differentiable})$$

Main Assumption:

$$|f_i'(x + h_i e_i) - f_i'(x)| \leq L_i |h_i|, \quad h_i \in R, \ i = 1, \dots, N,$$

where e_i is a coordinate vector. Then

$$f(x + h_i e_i) \leq f(x) + f_i'(x) h_i + \tfrac{L_i}{2} h_i^2. \quad x \in R^N, \ h_i \in R.$$

Define the coordinate steps: $T_i(x) \stackrel{\mathrm{def}}{=} x - \frac{1}{L_i} f_i'(x) e_i$. Then,

$$f(x) - f(T_i(x)) \quad \geq \quad \tfrac{1}{2L_i} [f_i'(x)]^2, \quad i = 1, \dots, N.$$

Random choice for coordinates

We need a special random counter $\mathcal{R}_\alpha$, $\alpha \in R$:

$$\textbf{Prob}\,[i] \;=\; p_\alpha^{(i)} \;=\; L_i^\alpha \cdot \left[\sum_{j=1}^{N} L_j^\alpha \right]^{-1}, \quad i = 1, \ldots, N.$$

Note: $\mathcal{R}_0$ generates uniform distribution.

```
Method $RCDM(\alpha, x_0)$

For $k \geq 0$ iterate:

1) Choose $i_k = \mathcal{R}_\alpha$.

2) Update $x_{k+1} = T_{i_k}(x_k)$.
```

Complexity bounds for RCDM

We need to introduce the following norms for $x, g \in R^N$:

$$\|x\|_\alpha = \left[\sum_{i=1}^{N} L_i^\alpha [x^{(i)}]^2 \right]^{1/2}, \quad \|g\|_\alpha^* = \left[\sum_{i=1}^{N} \frac{1}{L_i^\alpha} [g^{(i)}]^2 \right]^{1/2}.$$

After k iterations, $RCDM(\alpha, x_0)$ generates random output x_k, which depends on $\xi_k = \{i_0, \ldots, i_k\}$. Denote $\phi_k = E_{\xi_{k-1}} f(x_k)$.

Theorem. For any $k \geq 1$ we have

$$\phi_k - f^* \leq \frac{2}{k} \cdot \left[\sum_{j=1}^{N} L_j^\alpha \right] \cdot R_{1-\alpha}^2(x_0),$$

where $R_\beta(x_0) = \max\limits_{x} \left\{ \max\limits_{x_* \in X^*} \|x - x_*\|_\beta : \; f(x) \leq f(x_0) \right\}$.

Interpretation

Denote $S_\alpha = \sum\limits_{i=1}^{N} L_i^\alpha$.

1. $\alpha = 0$. Then $S_0 = N$, and we get

$$\phi_k - f^* \quad \leq \quad \frac{2N}{k} \cdot R_1^2(x_0).$$

Note

- We use the metric $\|x\|_1^2 = \sum\limits_{i=1}^{N} L_i [x^{(i)}]^2$.
- Matrix with diagonal $\{L_i\}_{i=1}^N$ can have its norm equal to n.
- Hence, for GM we can guarantee the same bound.

  But its cost of iteration is much higher!

Interpretation

2. $\alpha = \frac{1}{2}$. Denote

$$D_\infty(x_0) = \max_x \left\{ \max_{y \in X^*} \max_{1 \le i \le N} |x^{(i)} - y^{(i)}| : \; f(x) \le f(x_0) \right\}.$$

Then, $R^2_{1/2}(x_0) \le S_{1/2} D^2_\infty(x_0)$, and we obtain

$$\phi_k - f^* \quad \le \quad \frac{2}{k} \cdot \left[\sum_{i=1}^N L_i^{1/2} \right]^2 \cdot D^2_\infty(x_0).$$

Note:

- For the first order methods, the worst-case complexity of minimizing over a box depends on N.
- Since $S_{1/2}$ can be bounded, RCDM can be applied in situations when the usual GM fail.

Interpretation

3. $\alpha = 1$. Then $R_0(x_0)$ is the size of the initial level set in the standard Euclidean norm. Hence,

$$\phi_k - f^* \quad \leq \quad \tfrac{2}{k} \cdot \left[\sum_{i=1}^{N} L_i \right] \cdot R_0^2(x_0) \;\equiv\; \tfrac{2N}{k} \cdot \left[\tfrac{1}{N} \sum_{i=1}^{N} L_i \right] \cdot R_0^2(x_0).$$

Rate of convergence of GM can be estimated as

$$f(x_k) - f^* \leq \frac{\gamma}{k} R_0^2(x_0),$$

where γ satisfies condition $f''(x) \preceq \gamma \cdot I$, $x \in R^N$.
Note: maximal eigenvalue of symmetric matrix can reach its trace.

In the worst case, the rate of convergence of GM is the same as that of *RCDM*.

Minimizing the strongly convex functions

Theorem. Let $f(x)$ be strongly convex with respect to $\| \cdot \|_{1-\alpha}$ with convexity parameter $\sigma_{1-\alpha} > 0$.
Then, for $\{x_k\}$ generated by $RCDM(\alpha, x_0)$ we have

$$\phi_k - \phi^* \quad \leq \quad \left(1 - \frac{\sigma_{1-\alpha}}{S_\alpha}\right)^k (f(x_0) - f^*).$$

Proof: Let x_k be generated by $RCDM$ after k iterations.
Let us estimate the expected result of the next iteration.

$$\begin{aligned}
& f(x_k) - E_{i_k}(f(x_{k+1})) = \sum_{i=1}^{N} p_\alpha^{(i)} \cdot [f(x_k) - f(T_i(x_k))] \\
& \geq \sum_{i=1}^{N} \frac{p_\alpha^{(i)}}{2L_i} \left[f_i'(x_k)\right]^2 \ = \ \frac{1}{2S_\alpha} (\|f'(x_k)\|_{1-\alpha}^*)^2 \\
& \geq \frac{\sigma_{1-\alpha}}{S_\alpha} (f(x_k) - f^*).
\end{aligned}$$

It remains to compute expectation in ξ_{k-1}. $\square$

Confidence level of the answers

Note: We have proved that the expected values of random $f(x_k)$ are good.

Can we guarantee anything after a single run?

Confidence level: Probability $\beta \in (0,1)$, that some statement about random output is correct.

Main tool: Chebyschev inequality $\quad (\xi \geq 0)$:

$$\textbf{Prob}\,[\xi \geq T] \quad \leq \quad \frac{E(\xi)}{T}.$$

Our situation:

$$\textbf{Prob}\,[f(x_k) - f^* \geq \epsilon] \quad \leq \tfrac{1}{\epsilon}[\phi_k - f^*] \;\leq\; 1 - \beta.$$

We need $\phi_k - f^* \leq \epsilon \cdot (1-\beta)$. $\qquad$ Too expensive for $\beta \to 1$?

Regularization technique

Consider $f_\mu(x) = f(x) + \frac{\mu}{2}\|x - x_0\|_{1-\alpha}^2$. It is strongly convex.

Therefore, we can obtain $\phi_k - f_\mu^* \leq \epsilon \cdot (1-\beta)$ in

$$O\left(\frac{1}{\mu} S_\alpha \ln \frac{1}{\epsilon \cdot (1-\beta)}\right) \text{ iterations.}$$

Theorem. Define $\alpha = 1$, $\mu = \frac{\epsilon}{4R_0^2(x_0)}$, and choose

$$k \quad \geq \quad 1 + \frac{8S_1 R_0^2(x_0)}{\epsilon}\left[\ln \frac{2S_1 R_0^2(x_0)}{\epsilon} + \ln \frac{1}{1-\beta}\right].$$

Let x_k be generated by $RCDM(1, x_0)$ as applied to f_μ. Then

$$\mathbf{Prob}\left(f(x_k) - f^* \leq \epsilon\right) \quad \geq \quad \beta.$$

Note: $\beta = 1 - 10^{-p} \quad \Rightarrow \quad \ln 10^p = 2.3p.$

Implementation details: Random Counter

Given the values L_i, $i = 1, \dots, N$, generate efficiently random $i \in \{1, \dots, N\}$ with probabilities $\mathbf{Prob}\,[i = k] = L_k / \sum\limits_{j=1}^{N} L_j$.

Solution: a) Trivial $\Rightarrow$ $O(N)$ operations.

b). Assume $N = 2^p$. Define $p+1$ vectors $S_k \in R^{2^{p-k}}$, $k = 0, \dots, p$:

$$\begin{aligned} S_0^{(i)} &= L_i, \; i = 1, \dots, N. \\ S_k^{(i)} &= S_{k-1}^{(2i)} + S_{k-1}^{(2i-1)}, \; i = 1, \dots, 2^{p-k}, \quad k = 1, \dots, p. \end{aligned}$$

Algorithm: Make the choice in p steps, from top to bottom.

- If the element i of S_k is chosen, then choose in S_{k-1} either $2i$ or $2i-1$ in accordance to probabilities $\frac{S_{k-1}^{(2i)}}{S_k^{(i)}}$ or $\frac{S_{k-1}^{(2i-1)}}{S_k^{(i)}}$.

Difference: for $n = 2^{20} > 10^6$ we have $p = \log_2 N = 20$.

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. Sparsity coefficient: $\gamma(A) \overset{\text{def}}{=} \frac{p(A)}{MN}$.

Example 1: Matrix-vector multiplication

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

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 onto a 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 $y_+ \overset{\mathrm{def}}{=} Ax_+ = \underbrace{Ax}_{y} + Ad$.
- 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, $\kappa_A(d) \overset{\mathrm{def}}{=} \sum\limits_{j \in \sigma(d)} p(Ae_j)$, 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.

Examples

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 if a_i is sparse!

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

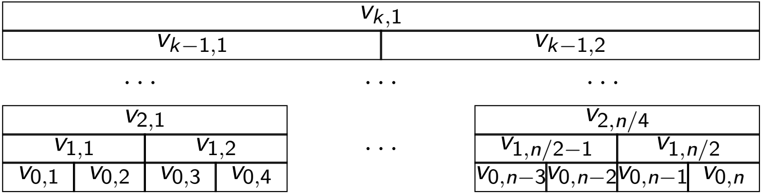
Fast updates in short computational trees

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: $v_{0,i} = x^{(i)}$, $i = 1, \ldots, n$. Size of the next level halves the size of the previous one:

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

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



$v_{k,1}$

$v_{k-1,1}$ $v_{k-1,2}$

$\ldots$ $\ldots$ $\ldots$

$v_{2,1}$ $\ldots$ $v_{2,n/4}$

$v_{1,1}$ $v_{1,2}$ $v_{1,n/2-1}$ $v_{1,n/2}$

$v_{0,1}$ $v_{0,2}$ $v_{0,3}$ $v_{0,4}$ $v_{0,n-3}$ $v_{0,n-2}$ $v_{0,n-1}$ $v_{0,n}$

Main advantages

- Important examples (symmetric functions)

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

- 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: $f^* \overset{\text{def}}{=} \min\limits_{x \in Q} f(x)$, where

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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!

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 $\kappa(A) \leq p^2$.

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:

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

Interpretation: Maximizing the self-esteem!
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, $\kappa(A) \approx p^2$.

Iteration Cost: $GM_s \approx p^2 \log_2 N, \quad GM \approx pN.$

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!