An introduction to nonsmooth convex optimization: numerical algorithms

Masoud Ahookhosh

Faculty of Mathematics, University of Vienna
Vienna, Austria

Convex Optimization I

January 29, 2014

Table of contents

Definition of problems

Definition 1 (Structural convex optimization).

Consider the following a convex optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array} \tag{1}$$

- $f(x)$ is a convex function;
- C is a closed convex subset of vector space V;

Properties:

- $f(x)$ can be smooth or nonsmooth;
- Solving nonsmooth convex optimization problems is much harder than solving differentiable ones;
- For some nonsmooth nonconvex cases, even finding a decent direction is not possible;
- The problem is involving linear operators.

Applications

Applications of convex optimization:

- Approximation and fitting;
  - Norm approximation;
  - Least-norm problems;
  - Regularized approximation;
  - Robust approximation;
  - Function fitting and interpolation;
- Statistical estimation;
  - Parametric and nonparametric distribution estimation;
  - Optimal detector design and hypothesis testing;
  - Chebyshev and Chernoff bounds;
  - Experiment design;
- Global optimization;
  - Find bounds on the optimal value;
  - Find approximation solutions;
  - Convex relaxation;

- Geometric problems;
  - Projection on and distance between sets;
  - Centering and classification;
  - Placement and location;
  - Smallest enclosed elipsoid;
- Image and signal processing;
  - Optimizing the number of image models using convex relaxation;
  - Image fusion for medical imaging;
  - Image reconstruction;
  - Sparse signal processing;
- Design and control of complex systems;
- Machine learning;
- Financial and mechanical engineering;
- Computational biology;

Dfinition: subgradient and subdifferential

Definition 2 (Subgradient and subdifferential).

- A vector $g \in \mathrm{R}^n$ is a subgradient of $f : \mathrm{R}^n \to \mathrm{R}$ at $x \in \mathrm{dom} f$ if

$$f(z) \geq f(x) + g^T(z - x), \tag{2}$$

  for all $z \in \mathrm{dom} f$.
- The set of all subgradients of f at x is called the subdifferential of f at x and denoted by $\partial f(x)$.

Definition 3 (Subdifferentiable functions).

- A function f is called subdifferentiable at x if there exists at least one subgradient of f at x.
- A function f is called subdifferentiable if it is subdifferentiable at all $x \in \mathrm{dom} f$.

Subgradient and subdifferential

Examples:

- if f is convex and differentiable, then the following first order condition holds:

$$f(z) \geq f(x) + \nabla f(x)^T (z - x), \tag{3}$$

  for all $z \in \operatorname{dom} f$. This implies: $\partial f(x) = \nabla f(x)$;

- Absolute value. Consider $f(x) = |x|$, then we have

$$\partial f(x) = \begin{cases} 1 & x > 0; \\ \text{[-1,1]} & x = 0; \\ -1 & x < 0. \end{cases}$$

  Thus, $g = \operatorname{sign}(x)$ is a subgradient of f at x.

Basic properties

Basic properties of subdifferential are as follows:

- The subdifferential $\partial f(x)$ is a closed convex set, even for a nonconvex function f.
- If f is convex and $x \in \mathrm{int}\ \mathrm{dom} f$, then $\partial f(x)$ is nonempty and bounded.
- $\partial(\alpha f(x)) = \alpha \partial f(x)$, for $\alpha \geq 0$.
- $\partial(\sum_{i=1}^{n} f_i(x)) = \sum_{i=1}^{n} \partial f_i(x)$.
- If $h(x) = f(Ax + b)$, then $\partial h(x) = A^T \partial f(Ax + b)$.
- If $h(x) = \max i = 1, \cdots, n f_i(x)$, then $\partial h(x) = \mathrm{conv} \bigcup \{\partial f_i(x) \mid f_i(x) = h(x) \quad i = 1, \cdots, n\}$.
- If $h(x) = \sup_{\beta} f_{\beta}(x)$, then $\partial h(x) = \mathrm{conv} \bigcup \{\partial f_{\beta}(x) \mid f_{\beta}(x) = h(x) \quad \beta \in B\}$.

How to calculate subgradients

Example: consider $f(x) = \|x\|_1 = \sum_{i=1}^{n} |x_i|$. It is clear that

$$f(x) = max\{s^T x \mid s_i \in \{-1, 1\}\}$$

We have $s^T x$ is differentiable and $g = \nabla f_i(x) = s$. Thus, for active $s^T x = \|x\|_1$, we should have

$$s_i = \begin{cases} 1 & s > 0; \\ \{\text{-1,1}\} & s = 0; \\ -1 & s < 0. \end{cases} \tag{4}$$

This clearly implies

$$\begin{aligned} \partial f(x) &= \operatorname{conv} \bigcup \{g \mid g \text{ of the form } (4), g^T x = \|x\|_1\} \\ &= \{g \mid \|g\|_\infty \leq 1, g^T x = \|x\|_1\}. \end{aligned}$$

Thus, $g = \operatorname{sign}(x)$ is a subgradient of f at x.

Optimality condition:

- First-order condition: A point x^* is a minimizer of a convex function f if and only if f is subdifferentiable at x^* and

$$0 \in \partial f(x^*), \tag{5}$$

  i.e., $g = 0$ is a subgradient of f at x^*.
- The condition (5) reduces to $\nabla f(x^*) = 0$ if f is differentiable at x^*.
- Analytical complexity: The number of calls of oracle, which is required to solve a problem up to the accuracy ε. This means the number of calls of oracle such that

$$f(x_k) - f(x^*) \leq \varepsilon; \tag{6}$$

- Arithmetical complexity: The total number of arithmetic operations, which is required to solve a problem up to the accuracy ε;

Numerical algorithms

The algorithms for solving nonsmooth convex optimization problems are commonly divided into the following classes:

- The nonsmooth balck-box optimization;
- Proximal mapping technique;
- Smoothing methods;

We here will not consider derivative-free and heuristic algorithms for solving nonsmooth convex optimization problems.

Nonsmooth black-box optimization: subgradient algorithms

The subgradient scheme for unconstrained problems:

$$x_{k+1} = x_k - \alpha_k g_k,$$

where g_k is a subgradient of the function f at x_k, and is a step size determined by:

- Constant step size: $\alpha_k = \alpha$;
- Constant step length: $\alpha_k = \gamma / \|g_k\|_2$;
- Square summable but not summable:
  $\alpha_k \geq 0, \ \sum_{k=1}^{n} = \alpha_k^2 < \infty, \ \sum_{k=1}^{n} = \alpha_k = \infty$;
- Nonsummable diminishing step size:
  $\alpha_k \geq 0, \ \lim_{k\to\infty} \alpha_k = 0, \ \sum_{k=1}^{n} = \alpha_k = \infty$;
- Nonsummable diminishing step length: $\alpha_k = \gamma_k / \|g_k\|$ such that
  $\gamma \geq 0, \ \lim_{k\to\infty} \gamma_k = 0, \ \sum_{k=1}^{n} = \gamma_k = \infty$.

The subgradient algorithm: properties

Main properties:

- The subgradient method is simple for implementations and applies directly to the nondifferentiable f;
- The step sizes are not chosen via line search, as in the ordinary gradient method;
- The step sizes are determined before running the algorithm and do not depend on any data computed during the algorithm;
- Unlike the ordinary gradient method, the subgradient method is not a descent method;
- The function vale is nonmonotone meaning that it can even increase;
- The subgradient algorithm is very slow for solving practical problems.

Bound on function values error:

If the Euclidean distance of the optimal set is bounded, $\|x_0 - x_*\|_2 \leq R$, and $\|g_k\|_2 \leq G$, then we have

$$f_k - f^* \leq \frac{R^2 + G^2 \sum_{i=1}^{k} \alpha_k^2}{2 \sum_{i=1}^{k} \alpha_k} := RHS. \tag{7}$$

- Constant step size: $k \to \infty \quad \Rightarrow \quad RHS \to G^2\alpha/2$;
- Constant step length: $k \to \infty \quad \Rightarrow \quad RHS \to G\gamma/2$;
- Square summable but not summable: $k \to \infty \quad \Rightarrow \quad RHS \to 0$;
- Nonsummable diminishing step size: $k \to \infty \quad \Rightarrow \quad RHS \to 0$;
- Nonsummable diminishing step length: $k \to \infty \quad \Rightarrow \quad RHS \to 0$.

Example: we now consider the LASSO problem

$$\mathrm{minimize}_{x \in \mathbb{R}^n} \quad \frac{1}{2}\|Ax - b\|_2^2 + \|x\|_1, \tag{8}$$

where A and b are randomly generated.

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda\|x\|_1$

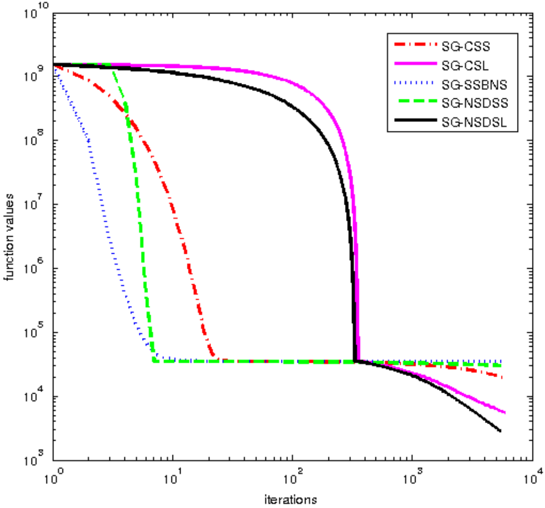


Figure 1: A comparison among the subgradient algorithms when they stopped after 60 seconds of the running time (dense, $m = 2000$ and $n = 5000$)

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda\|x\|_1$

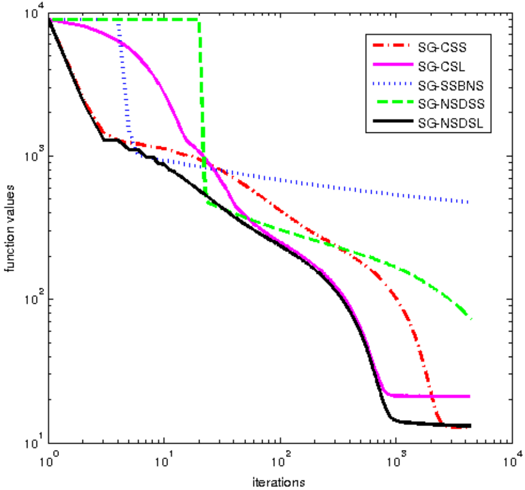


Figure 2: A comparison among the subgradient algorithms when they stopped after 20 seconds of the running time (sparse, $m = 2000$ and $n = 5000$)

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda \|x\|_2^2$

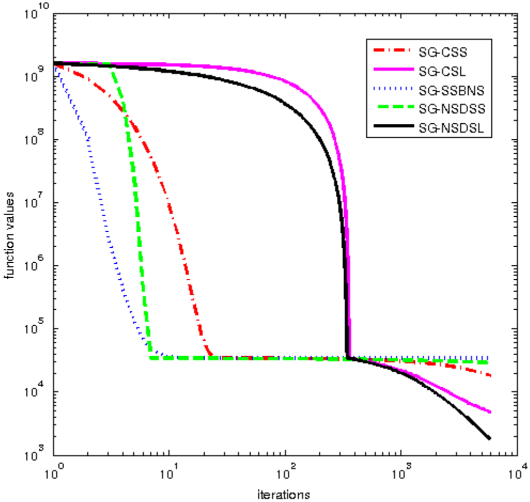


Figure 3: A comparison among the subgradient algorithms when they stopped after 60 seconds of the running time (dense, $m = 2000$ and $n = 5000$)

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda \|x\|_2^2$

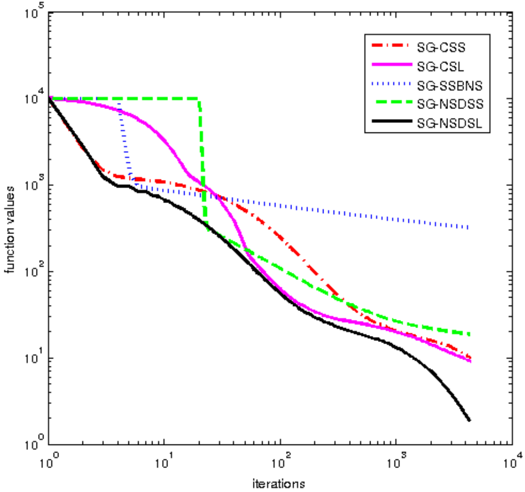


Figure 4: A comparison among the subgradient algorithms when they stopped after 20 seconds of the running time (sparse, $m = 2000$ and $n = 5000$)

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda\|x\|_1$

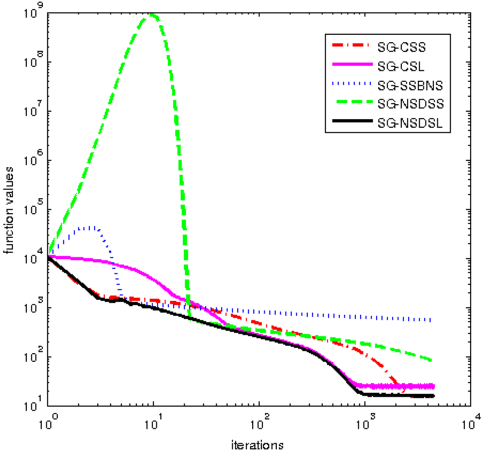


Figure 5: The nonmonotone behaviour of the original subgradient algorithms when they stopped after 20 seconds of the running time (sparse, $m = 2000$ and $n = 5000$)

Projected subgradient algorithm

Consider the following constrained problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C, \end{array} \tag{9}$$

where C is a simple convex set. Then the projected subgradient scheme is given by

$$x_{k+1} = P(x_k - \alpha_k g_k), \tag{10}$$

where

$$P(y) = \text{argmin}_{x \in C} \frac{1}{2} \|x - y\|_2^2. \tag{11}$$

- Nonnegative orthant;
- Affine set;
- Box or unit ball;
- Unit simplex;
- An ellipsoid;
- Second-order cone;
- Positive semidefinite cone;

Projected subgradient algorithm

Example: Let us to consider

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & Ax = b, \end{array} \tag{12}$$

where $x \in \mathtt{R}^n$, $x \in \mathtt{R}^m$ and $A \in \mathtt{R}^{m\times n}$. Considering the set $C = \{x \mid Ax = b\}$, we have

$$P(y) = y - A^T(AA^T)^{-1}(Ay - b). \tag{13}$$

The projected subgradient algorithm can be summarized as follows

$$x_{k+1} = x_k - \alpha_k(I - A^T(AA^T)^{-1}A)g_k. \tag{14}$$

By setting $g_k = \mathrm{sign}(x_k)$, we obtain

$$x_{k+1} = x_k - \alpha_k(I - A^T(AA^T)^{-1}A)\mathrm{sign}(x_k). \tag{15}$$

Proximal gradient algorithm

Consider a composite function as follows

$$h(x) = f(x) + g(x). \tag{16}$$

Characteristics of the considered convex optimization:

- Appearing in many applications in science and technology: signal and image processing, machine learning, statistics, inverse problems, geophysics and so on.
- In convex optimization $\longrightarrow$ every local optimum is global optimizer.
- Most of the problems are combination of both smooth and nonsmooth functions:

  $$h(x) = f(Ax) + g(Bx),$$

  where $f(Ax)$ and $g(Ax)$ are respectively smooth and nonsmooth functions.
- Function and subgradient evaluations are so costly: Affine transformations are the most costly part of the computation.
- They are involving high-dimensional data.

Proximal gradient algorithm

The algorithm involve two step, namely forward and backward, as follows:

Algorithm 1: PGA proximal gradient algorithm

Input: $\alpha_0 \in (0, 1]$; y_0; $\epsilon > 0$;

begin

  while *stopping criteria are not hold* **do**

    $y_{k+1} = x_k - \alpha_k g_k$;

    $x_{k+1} = \text{argmin}_{x \in \mathbb{R}^n} \frac{1}{2}\|x - y_{k+1}\|_2^2 + g(x)$;

  end

end

- First step called forward because it aims to go toward the minimizer, and the second step called backward step because it remind us feasibility step of the projected gradient method.
- It is clear that the projected gradient method is a spacial case of PGA.

Smoothing algorithms

The smoothing algorithms involve the following steps:

- Reformulate the problem in the appropriate form for smoothing processes;
- Make the problem smooth;
- Solve the problem with smooth convex solvers.

Nesterov's smoothing algorithm:

- Reformulate the problem in the form of the minimax problem (saddle point representation);
- Add a strongly convex prox function to the reformulated problem to make it smooth;
- Solve the problem with optimal first-order algorithms.

Optimal complexity for first-order methods

Nemirovski and Yudin in 1983 proved the following complexity bound for smooth and nonsmooth problems:

Theorem 4 (Complexity analysis).

Suppose that f is a convex function. Then complexity bounds for smooth and nonsmooth problems are

- *(Nonsmooth complexity bound) If the point generated by the algorithm stays in bounded region of the interior of C, or f is Lipschitz continuous in C, then the total number of iterations needed is $O\left(\frac{1}{\epsilon^2}\right)$. Thus the asymptotic worst case complexity is $O\left(\frac{1}{\epsilon^2}\right)$.*
- *(Smooth complexity bound) If f has Lipschitz continuous gradient, the total number of iterations needed for the algorithm is $O\left(\frac{1}{\sqrt{\epsilon}}\right)$.*

Optimal first-order algorithms

Some popular optiml first-order algorithms:

- Nonsummable diminishing subgradient algorithm;
- Nesterov's 1983 smooth algorithm;
- Nesterov and Nemiroski's 1988 smooth algorithm;
- Nesterov's constant step algorithm;
- Nesterov's 2005 smooth algorithm;
- Nesterov's composite algorithm;
- Nesterov's universal gradient algorithm;
- Fast iterative shrinkage-thresholding algorithm
- Tseng's 2008 single projection algorithm;
- Lan's 2013 bundle-level algorithm;
- Neumaier's 2014 fast subgradient algorithm;

Algorithm 2: NES83 Nesterov's 1983 algorithm

Input: select z such that $z \neq y_0$ and $g_{y_0} \neq g_z$; y_0; $\epsilon > 0$;

begin

$a_0 \leftarrow 0$; $x_{-1} \leftarrow y_0$;

$\alpha_{-1} \leftarrow \|y_0 - z\| / \|g_{y_0} - g_z\|$;

while *stopping criteria are not hold* **do**

$\hat{\alpha}_k \leftarrow \alpha_{k-1}$; $\hat{x}_k \leftarrow y_k - \hat{\alpha}_k g_{y_k}$;

while $f(\hat{x}_k) < f(y_k) - \frac{1}{2}\hat{\alpha}_k \|g_{y_k}\|^2$ **do**

$\hat{\alpha}_k \leftarrow \rho\hat{\alpha}_k$; $\hat{x}_k \leftarrow y_k - \hat{\alpha}_k g_{y_k}$;

end

$x_{k+1} \leftarrow \hat{x}_k$; $\alpha_k \leftarrow \hat{\alpha}_k$;

$a_{k+1} \leftarrow \left(1 + \sqrt{4a_k^2 + 1}\right)/2$;

$y_{k+1} \leftarrow x_k + (a_k - 1)(x_k - x_{k-1})/a_{k+1}$;

end

end

Algorithm 3: FISTA fast iterative shrinkage-thresholding algorithm

Input: select z such that $z \neq y_0$ and $g_{y_0} \neq g_z$; y_0; $\epsilon > 0$;

begin

  while *stopping criteria are not hold* **do**

    $\alpha_k \leftarrow 1/L$;

    $z_k \leftarrow y_k - \alpha_k g_{y_k}$;

    $x_k = \mathrm{argmin}_x \frac{L}{2}\|x - z_k\|_2^2 + g(x)$;

    $a_{k+1} \leftarrow \left(1 + \sqrt{4a_k^2 + 1}\right)/2$;

    $y_{k+1} \leftarrow x_k + (a_k - 1)(x_k - x_{k-1})/a_{k+1}$;

  end

end

By this adaptation, FISTA obtains the optimal complexity of smooth first-order algorithms

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda\|x\|_1$

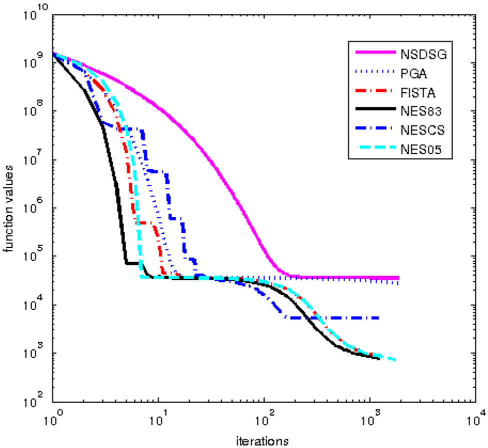


Figure 6: A comparison among the subgradient algorithms when they stopped after 60 seconds of the running time (dense, $m = 2000$ and $n = 5000$)

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda\|x\|_1$

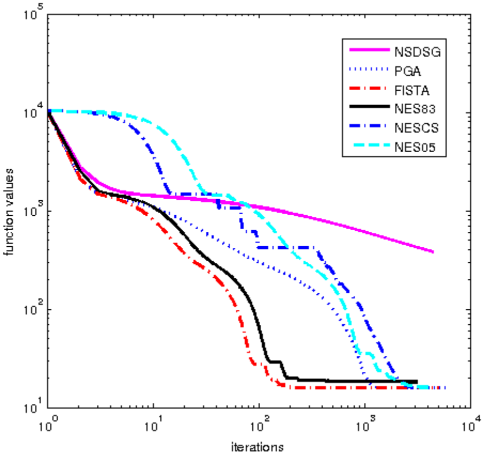


Figure 7: A comparison among the subgradient algorithms when they stopped after 20 seconds of the running time (sparse, $m = 2000$ and $n = 5000$)

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda \|x\|_2^2$

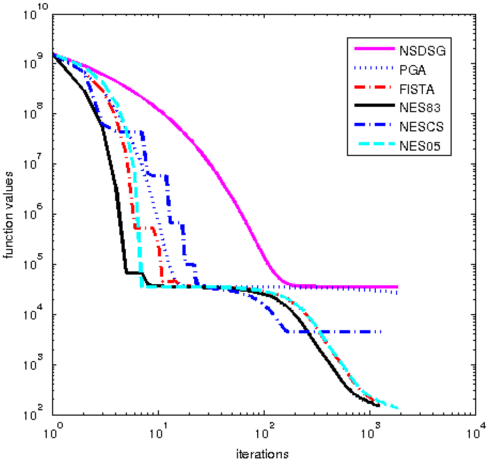


Figure 8: A comparison among the subgradient algorithms when they stopped after 60 seconds of the running time (dense, $m = 2000$ and $n = 5000$)

Numerical experiment: $f(x) = \|Ax - b\|_2^2 + \lambda\|x\|_2^2$

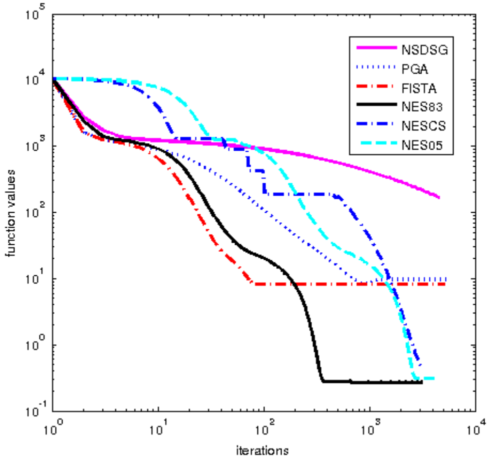


Figure 9: A comparison among the subgradient algorithms when they stopped after 20 seconds of the running time (sparse, $m = 2000$ and $n = 5000$)

Conclusions

Summarizing our discussion:

- They are appearing in applications much more than smooth optimization;
- Solving nonsmooth optimization problems is much harder than common smooth optimization;
- The most efficient algorithms for solving them are first-order methods;
- There are no normal stopping criterion in corresponding algorithms;
- The algorithms are divided into three classes:
  - Nonsmooth back-box algorithms;
  - Proximal mapping algorithms;
  - Smoothing algorithms;
- Analytical complexity of the algorithms is the most important part of theoretical results;
- Optimal complexity algorithms are so efficient to solve practical problems.

References

- [1] Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM Journal on Imaging Sciences, 2 (2009), 183–202.
- [2]Boyd, S., Xiao, L., Mutapcic, A.: Subgradient methods, (2003).
- [3] Nemirovski, A.S., Yudin, D.: Problem Complexity and Method Efficiency in Optimization. Wiley- Interscience Series in Discrete Mathematics. Wiley, XV (1983).
- [4] Nesterov, Y.E.: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer, Massachusetts (2004).
- [5] Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$, Doklady AN SSSR (In Russian), 269 (1983), 543-547. English translation: Soviet Math. Dokl. 27 (1983), 372–376.

Thank you for your consideration

Optimal subgradient methods for large-scale convex optimization

Masoud Ahookhosh

Faculty of Mathematics, University of Vienna
Vienna, Austria

Convex Optimization I

January 30, 2014

Table of contents

Definition of problems

Definition 1 (Structural convex optimization).

Consider the following a convex optimization problem

$$\begin{aligned} &\text{minimize} \quad && f(x) \\ &\text{subject to} \quad && x \in C \end{aligned} \tag{1}$$

- $f(x)$ is a convex function;
- C is a closed convex subset of vector space V;

Properties:

- $f(x)$ can be smooth or nonsmooth;
- Solving nonsmooth convex optimization problems is much harder than solving differentiable ones;
- For some nonsmooth nonconvex cases, even finding a decent direction is not possible;
- The problem is involving linear operators.

Which kind of algorithms can deal with these problems?

Appropriate algorithms for this class of problems: First-order methods

- Gradient and Subgradient projection algorithms;
- Conjugate gradient algorithms;
- Optimal gradient and subgradient algorithms;
- Proximal mapping and Soft-thresholding algorithms;

Optimal complexity for COP (Nemirovski and Yudin 1983):

- Smooth problems $\rightarrow O\left(\frac{1}{\sqrt{\epsilon}}\right)$.
- Nonsmooth problems $\rightarrow O\left(\frac{1}{\epsilon^2}\right)$.

Some examples:

- N83: Nesterovs single-projection (1983);
- N07: Nesterovs dual-projection (2007);
- FISTA: Beck and Teboulle optimal proximal algorithm (2009);
- N07: Nesterovs universal gradient (2013);
- OSGA & ASGA: Ahookhosh and Neumaier affine subgradient (2013).

Optimal SubGradient Algorithm (OSGA): Motivation

The primary aim:

$$0 \leq f(x_b) - f(x^*) \leq \text{Bound} \to 0 \tag{2}$$

To do so, we consider:

- **First-order oracle:** black-box unit that computes $f(x)$ and $\nabla f(x)$ for the numerical method at each point x:

$$\mathcal{O}(x) = (f(x), \nabla f(x)). \tag{3}$$

- **Linear relaxation:** $f(z) \geq \gamma + \langle h, z \rangle$
- **Prox function:** Q is continuously differentiable, $Q_0 = \inf_{z \in C} Q(z) > 0$ and

$$Q(z) \geq Q(x) + \langle q_Q(x), z - x \rangle + \frac{1}{2}\|z - x\|^2, \forall x, z \in C. \tag{4}$$

- **Auxiliary subproblem:**

$$E(\gamma, h) = \inf_{z \in C} \frac{\gamma + \langle h, z \rangle}{Q(z)} \tag{5}$$

where $z = U(\gamma, h) \in C$ and $E(\gamma, h)$ and $U(\gamma, h)$ are computable.

- **Error bound:** from the definition of $E(\gamma, h)$, the linear relaxation and some manipulations, it can be concluded

$$0 \leq f(x_b) - f(x^*) \leq \eta Q(x^*). \tag{6}$$

- **How to use in algorithm:**
  - If $Q(x^*)$ is computable, then the error bound $\eta Q(x^*)$ is appliable.
  - Otherwise, we will search for decreasing $\{\eta_k\}$ satisfying

$$0 \leq f(x_b) - f(x^*) \leq \epsilon Q(x^*). \tag{7}$$

for some constant $\epsilon > 0$.

Algorithmic structure

Algoritm 2: Optimal SubGradient Algorithm (OSGA)

Input: λ, $\alpha_{max} \in (0\ 1)$, $0 < \kappa' \leq \kappa$, $\mu \geq 0$, $\epsilon > 0$ and f_{target}.

Begin

Choose x_b; Stop if $f(x_b) \leq f_{target}$;

$h = g(x_b)$; $\gamma = f(x_b) - \langle h, x_b \rangle$;

$\gamma_b = \gamma - f(x_b)$; $u = U(\gamma_b, h)$; $\eta = E(\gamma_b, h) - \mu$; α_{max};

While $\eta > \epsilon$

$x = x_b + \alpha(u - x_b)$; $g = g(x)$; $\bar{h} = h + \alpha(g - h)$;

$\bar{\gamma} = \gamma + \alpha(f(x) + \langle g, x \rangle - \gamma)$; $x_b' = \mathrm{argmin}\{f(x_b), f(x)\}$;

$\gamma_b' = \bar{\gamma} - f(x_b')$; $u' = U(\gamma_b', \bar{h})$; $x' = x_b + \alpha(u' - x_b)$;

Choose $\bar{x}_b = \mathrm{argmin}\{f(x_b'), f(x')\}$;

$\bar{\gamma}_b = \bar{\gamma} - f(\bar{x}_b)$; $u' = U(\bar{\gamma}_b, \bar{h})$; $\eta = E(\bar{\gamma}_b, \bar{h}) - \mu$;

$x_b = \bar{x}_b$; Stop if $f(x_b) \leq f_{target}$;

Update $\alpha, h, \gamma, \eta, u$;

End

End

Theoretical Analysis

Theorem 2 (Complexity analysis).

Suppose that f is a convex function. Then complexity bounds for smooth and nonsmooth problems are

- *(Nonsmooth complexity bound) If the point generated by Algorithm 2 stay in bounded region of the interior of C, or f is Lipschitz continuous in C, then the total number of iterations needed is $O\left(\frac{1}{\epsilon^2}\right)$. Thus the asymptotic worst case complexity is $O\left(\frac{1}{\epsilon^2}\right)$.*
- *(Smooth complexity bound) If f has Lipschitz continuous gradient, the total number of iterations needed for the algorithm is $O\left(\frac{1}{\sqrt{\epsilon}}\right)$.*

$\Rightarrow$ OSGA IS AN OPTIMAL METHOD

Prox function and subproblem solving

- Quadratic norm:

$$\|z\| := \sqrt{\langle Bz, z\rangle}$$

- Dual norm:

$$\|h\|_* := \|B^{-1}h\| = \sqrt{\langle h, B^{-1}h\rangle}$$

- Prox function:

$$Q(z) := Q_0 + \frac{1}{2}\|z - z_0\|^2$$

- Subproblem solution:

$$U(\gamma, h) = z_0 - E(\gamma, h)^{-1}B^{-1}h$$

- $E(\gamma, h) = \frac{-\beta+\sqrt{\beta^2+2Q_0\|h\|_*^2}}{2Q_0} = \frac{\|h\|_*^2}{\beta+\sqrt{\beta^2+2Q_0\|h\|_*^2}}.$

Numerical experiments: linear inverse problem

Definition 3 (Linear inverse problem).

We consider the following convex optimization problems:

$$Ax = b + \delta \tag{8}$$

- $A \in R^{m \times n}$ is a matrix or a linear operator, $x \in R^n$ and $b, \delta \in R^m$

Examples:

- Signal and image processing
- Machine learning and statistics
- Compressed sensing
- Geophysics
- ...

Approximate solution

Definition 4 (Least square problem).

$$\text{Minimize} \quad \frac{1}{2}\|Ax - b\|_2^2 \tag{9}$$

- The problem includes high-dimensional data
- The problem is usually ill-conditioned and singular

Alternative problems: Tikhonov regularization:

$$\text{minimize} \quad \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_2^2. \tag{10}$$

General case:

$$\text{minimize} \quad \frac{1}{2}\|Ax - b\|_2^2 + \lambda g(x), \tag{11}$$

where $g(x)$ is a regularization term like $g(x) = \|x\|_p$ for $p \geq 1$ or $0 \leq p < 1$ and $g(x) = \|x\|_{ITV}$ or $\|x\|_{ATV}$.

Isotropic and anisotropic total variation

Two standard choices of discrete TV-based regularizers, namely **isotropic total variation** and **anisotropic total variation**, are popular in signal and image processing, where they are respectively defined by

$$\begin{aligned}\|X\|_{ITV} &= \sum_{i}^{m-1}\sum_{j}^{n-1}\sqrt{(X_{i+1,j}-X_{i,j})^2+(X_{i,j+1}-X_{i,j})^2} \\ &\quad + \sum_{i}^{m-1}|X_{i+1,n}-X_{i,n}| + \sum_{i}^{n-1}|X_{m,j+1}-X_{m,j}|,\end{aligned} \tag{12}$$

and

$$\begin{aligned}\|X\|_{ATV} &= \sum_{i}^{m-1}\sum_{j}^{n-1}\{|X_{i+1,j}-X_{i,j}|+|X_{i,j+1}-X_{i,j}|\} \\ &\quad + \sum_{i}^{m-1}|X_{i+1,n}-X_{i,n}| + \sum_{i}^{n-1}|X_{m,j+1}-X_{m,j}|,\end{aligned} \tag{13}$$

where $X \in \mathbb{R}^{m\times n}$.

Denising of the noisy image

(a) Original image

(b) Noisy image

Denising by solving $\min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_{ITV}$

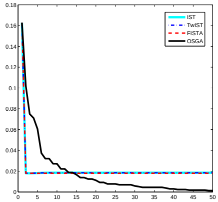
(c) OSGA

(d) IST

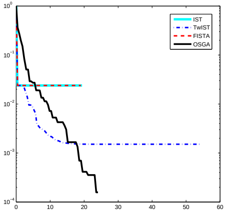
(e) TwIST

(f) FISTA

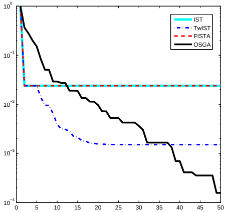
Denising by solving $\min_x \quad \frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_{ITV}$



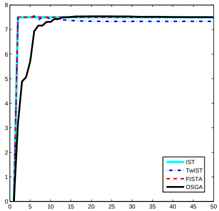
(a) step vs. iter



(b) Func vs. time



(c) Func vs. iter



(d) ISNR vs. iter

Inpainting images with missing data

(a) Original image

(b) Noisy image

Inpainting by solving $\min_x \quad \frac{1}{2}\|Ax-b\|_2^2+\lambda\|x\|_{ITV}$

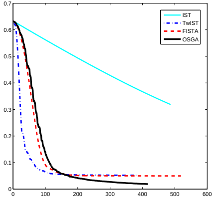
(c) OSGA

(d) IST

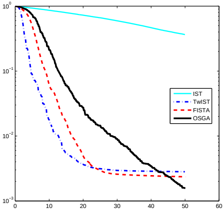
(e) TwIST

(f) FISTA

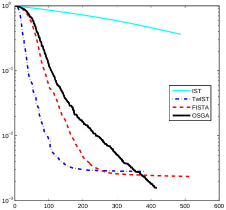
Inpainting by solving $\min_x \quad \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_{ITV}$



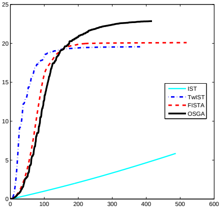
(a) step vs. iter



(b) Func vs. time



(c) Func vs. iter



(d) ISNR vs. iter

Deblurring of the blurred/noisy image

(a) Original image

(b) Noisy image

Deblurring by solving $\min_x \quad \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_{ITV}$

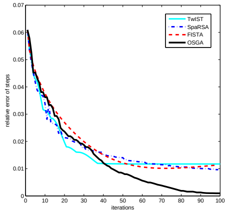
(c) OSGA

(d) IST

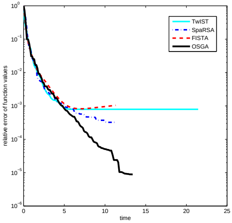
(e) TwIST

(f) FISTA

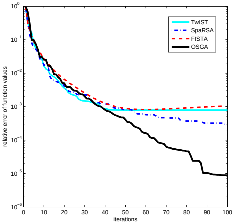
Deblurring by solving $\min_x \quad \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_{ITV}$



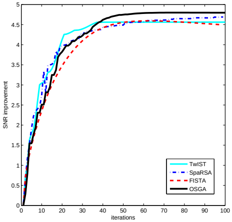
(a) step vs. iter



(b) Func vs. time

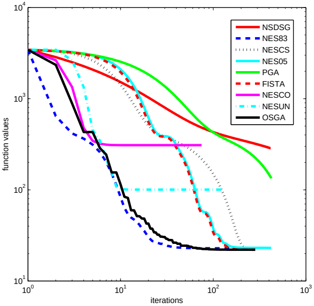


(c) Func vs. iter

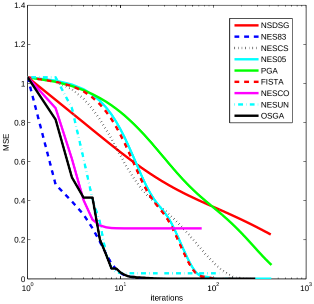


(d) ISNR vs. iter

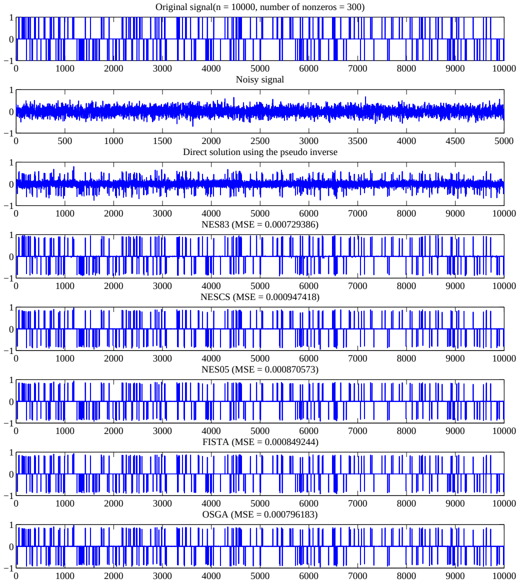
A comparison among first-order methods for a sparse signal recovery by solving $\min_x \quad \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1$



(a) step vs. iter



(b) Func vs. time


Original signal(n = 10000, number of nonzeros = 300)
Noisy signal
Direct solution using the pseudo inverse
NES83 (MSE = 0.000729386)
NESCS (MSE = 0.000947418)
NES05 (MSE = 0.000870573)
FISTA (MSE = 0.000849244)
OSGA (MSE = 0.000796183)

Conclusions and references

Summarizing our discussion:

- OSGA is optimal algorithms for both smooth and nonsmooth convex optimization problems;
- OSGA is feasible and avoid using the Lipschitz information;
- Low memory requirement OSGA makes them to be appropriate for solving high-dimensional problems;
- OSGA is efficient and robust in applications and practice and superior to some state-of-the-art solvers.

References

- [1] A. Neumaier, OSGA: fast subgradient algorithm with optimal complexity, *Manuscript*, University of Vienna, 2014.
- [5] M. Ahookhosh, A. Neumaier, Optimal subgradient methods with application in large-scale linear inverse problems, *Manuscript*, University of Vienna, 2014.
- [3] M. Ahookhosh, A. Neumaier, Optimal subgradient-based methods for convex constrained optimization I: theoretical results, *Manuscript*, University of Vienna, 2014.
- [4] M. Ahookhosh, A. Neumaier, Optimal subgradient-based methods for convex constrained optimization II: numerical results, *Manuscript*, University of Vienna, 2014.

Thank you for your consideration