Algorithms I

Pontus Giselsson

Today's lecture

- optimality conditions
- subgradient method
- gradient method
- proximal point method (resolvent method)
- forward-backward splitting

Optimality conditions

- assume f, g proper closed and convex, L linear operator
- we want to solve

$$\begin{array}{ll} \text{minimize} & f(x) + g(y) \\ \text{subject to} & Lx = y \end{array}$$

- optimality condition (Fermat's rule)

$$0 \in \partial f(x) + \partial (g \circ L)(x)$$

- optimality condition of dual

$$0 \in \partial (f^* \circ -L^*)(\mu) + \partial g^*(\mu)$$

- both can be written as sum of maximally monotone operators

Optimality conditions cont'd

- the condition $0 \in \partial f(x) + \partial(g \circ L)(x)$ can be written as

$$
\begin{aligned}
0 &\in \partial f(x) + L^*\mu \\
0 &\in \partial g(y) - \mu \\
0 &= Lx - y
\end{aligned}
$$

- or

$$
\begin{aligned}
0 &\in \partial f(x) + L^*\mu \\
0 &\in \partial g^*(\mu) - Lx
\end{aligned}
$$

Optimality conditions cont'd

- let

$$F(x,\mu) = (\partial f(x), \partial g^*(\mu)), \qquad M(x,\mu) = (L^*\mu, -Lx)$$

- F, M maximally monotone (M skew symmetric, i.e. $M^* = -M$)
- consider the optimality condition

$$\begin{aligned} 0 &\in \partial f(x) + L^*\mu \\ 0 &\in \partial g^*(\mu) - Lx \end{aligned}$$

- it can be written as

$$0 \in F(x,\mu) + M(x,\mu)$$

i.e., sum of two maximal monotone operators

Sums of several functions

- assume f_1, f_2, g proper closed and convex, L_1, L_2 linear operators
- we want to solve

$$\begin{array}{ll} \text{minimize} & f_1(x) + f_2(y) + g(z) \\ \text{subject to} & L_1 x + L_2 y = z \end{array}$$

- let $f(x, y) = f_1(x) + f_2(y)$ and $L(x, y) = L_1 x + L_2 y$
- then problem is

$$\text{minimize } f(x, y) + g(L(x, y))$$

- obviously more f_i functions can be added

Sums of several functions

- assume f, g_1, g_2 proper closed and convex, L_1, L_2 linear operators
- we want to solve

$$\begin{array}{ll} \text{minimize} & f(x) + g_1(y) + g_2(z) \\ \text{subject to} & L_1 x = y \\ & L_2 x = z \end{array}$$

- let $g(y,z) = g_1(y) + g_2(z)$ and $L(x) = (L_1 x, L_2 x)$
- then problem is

$$\text{minimize } f(x) + g(Lx)$$

- obviously more g_i functions can be added

Monotone inclusion problems

- optimality conditions is sum of maximally monotone operators

$$0 \in Ax + Bx$$

  for different A and B

- consider the more general formulation

$$0 \in Ax + L^* B(Lx)$$

- inclusion holds if and only if

$$\begin{array}{l} 0 \in Ax + L^*\mu \\ 0 \in B(Lx) - \mu \end{array} \quad \Leftrightarrow \quad \begin{array}{l} 0 \in Ax + L^*\mu \\ 0 \in B^{-1}\mu - Lx \end{array}$$

- let $F(x,\mu) = (Ax, B^{-1}\mu)$ and $M(x,\mu) = (L^*\mu, -Lx)$
- condition is sum of maximally monotone operators

$$0 \in F(x,\mu) + M(x,\mu)$$

How to develop an algorithm

- write optimality condition as fixed-point to some operator
- show convergence properties when iterating operator

Subgradient method

- assume f is closed and convex
- optimality condition

$$x = \underset{x}{\operatorname{argmin}}\, f(x) \quad \Leftrightarrow \quad 0 \in \partial f(x) \quad \Leftrightarrow \quad x \in x - \gamma \partial f(x)$$

- algorithm:

$$x^{k+1} = x^k - \gamma \partial f(x^k)$$

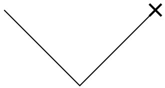
- if we find a fixed-point, we solve the problem
- does it converge to a fixed-point?

Example

- consider minimizing the function $f(x) = |x|$:
- let $\gamma = c$:
- iteration if $x^k \neq nc$ where $n = \ldots, -1, 0, 1, \ldots$:

$$x^{k+1} = x^k - c\,\mathrm{sign}(x)$$

- will jump back and forth over optimal point



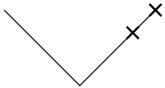
- fixed step-size does not work

Example

- consider minimizing the function $f(x) = |x|$:
- let $\gamma = c$:
- iteration if $x^k \neq nc$ where $n = \ldots, -1, 0, 1, \ldots$:

$$x^{k+1} = x^k - c\,\mathrm{sign}(x)$$

- will jump back and forth over optimal point



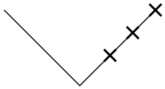
- fixed step-size does not work

Example

- consider minimizing the function $f(x) = |x|$:
- let $\gamma = c$:
- iteration if $x^k \neq nc$ where $n = \dots, -1, 0, 1, \dots$:

$$x^{k+1} = x^k - c\,\mathrm{sign}(x)$$

- will jump back and forth over optimal point



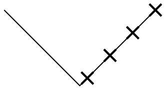
- fixed step-size does not work

Example

- consider minimizing the function $f(x) = |x|$:
- let $\gamma = c$:
- iteration if $x^k \neq nc$ where $n = \ldots, -1, 0, 1, \ldots$:

$$x^{k+1} = x^k - c\,\mathrm{sign}(x)$$

- will jump back and forth over optimal point



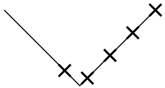
- fixed step-size does not work

Example

- consider minimizing the function $f(x) = |x|$:
- let $\gamma = c$:
- iteration if $x^k \neq nc$ where $n = \ldots, -1, 0, 1, \ldots$:

$$x^{k+1} = x^k - c\,\text{sign}(x)$$

- will jump back and forth over optimal point



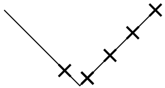
- fixed step-size does not work

Example

- consider minimizing the function $f(x) = |x|$:
- let $\gamma = c$:
- iteration if $x^k \neq nc$ where $n = \ldots, -1, 0, 1, \ldots$:

$$x^{k+1} = x^k - c\,\mathrm{sign}(x)$$

- will jump back and forth over optimal point



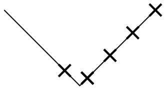
- fixed step-size does not work

Example

- consider minimizing the function $f(x) = |x|$:
- let $\gamma = c$:
- iteration if $x^k \neq nc$ where $n = \ldots, -1, 0, 1, \ldots$:

$$x^{k+1} = x^k - c\,\mathrm{sign}(x)$$

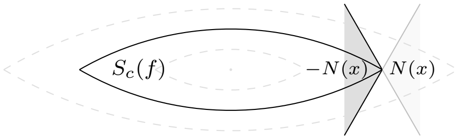
- will jump back and forth over optimal point



- fixed step-size does not work

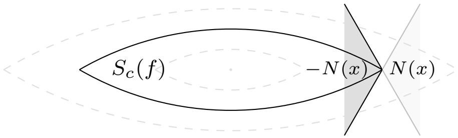
Not descent method

- a (-) subgradient does not necessarily specify a descent direction
- subgradient is in normal cone to level set:

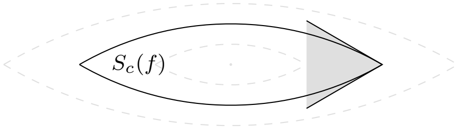


Not descent method

- a (-) subgradient does not necessarily specify a descent direction
- subgradient is in normal cone to level set:



- would want to find element in tangent cone to get descent



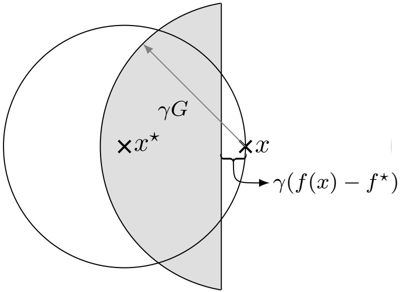
- such elements hard to compute

Graphical interpretation of convergence

- assume that $u \in \partial f(x)$ and $\partial f(x) \subseteq B_G(x)$ for all x
- subdifferential definition $f^\star := f(x^\star) \geq f(x) + \langle u, x^\star - x\rangle$ implies

$$\langle u, x - x^\star\rangle \geq f(x) - f^\star \geq 0$$

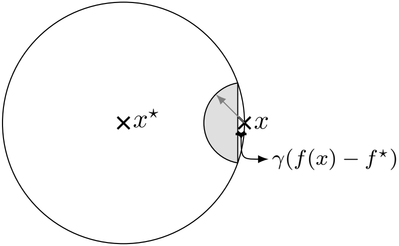
- $x - \gamma u$ can end up in gray region:



- half-circle due to $\gamma\partial f(x) \subseteq B_{\gamma G}(x)$
- vertical line due to scalar-product inequality (left of x if $f(x) > f(x^\star)$)

Graphical interpretation of convergence

- if γ small enough, $x - \gamma u$ ends up somewhere in gray region:



- i.e., the distance to the fixed-point is decreased
- this γ value is not known a priori
- it depends on $f(x) - f^\star \Rightarrow$ diminishing step-size

Convergence

- let $u \in \partial f(x^k)$ and $\partial f(x) \subseteq B_G(0)$ for all x
- recall used subgradient definition:

$$f^\star = f(x^\star) \geq f(x^k) + \langle u, x^\star - x^k \rangle$$

- then

$$\begin{aligned}
\|x^{k+1} - x^\star\|^2 &= \|x^k - \gamma_k u - x^\star\|^2 \\
&= \|x^k - x^\star\|^2 - 2\gamma_k \langle u, x^k - x^\star \rangle + \gamma_k^2 \|u\|^2 \\
&\leq \|x^k - x^\star\|^2 - 2\gamma_k (f(x^k) - f^\star) + \gamma_k^2 G^2
\end{aligned}$$

Convergence

- apply recursively up to $k = n$ to get

$$(0 \leq)\|x^{n+1} - x^\star\|^2 \leq \|x^0 - x^\star\|^2 - 2\sum_{k=0}^{n} \gamma_k(f(x^k) - f^\star) + G^2 \sum_{k=0}^{n} \gamma_k^2$$

- let $f_{\mathrm{best}}^n = \min_{k=1,\ldots,n} f(x^k)$, since $f(x^k) \geq f^\star$, we have

$$(f_{\mathrm{best}}^n - f^\star)\sum_{i=0}^{n} \gamma_k = \sum_{i=0}^{n} \gamma_k(f_{\mathrm{best}}^n - f^\star) \leq \sum_{k=0}^{n} \gamma_k(f(x^k) - f^\star)$$

- therefore

$$f_{\mathrm{best}} - f^\star \leq \frac{\|x^0 - x^\star\|^2 + G^2 \sum_{k=0}^{n} \gamma_k^2}{2\sum_{k=0}^{n} \gamma_k}$$

Step-size requirements

- under what conditions of γ_k do we get convergence?

$$f_{\text{best}} - f^\star \le \frac{\|x^0 - x^\star\|^2 + G^2 \sum_{k=0}^n \gamma_k^2}{2 \sum_{k=0}^n \gamma_k}$$

- if, for instance,

$$\sum_{k=0}^{\infty} \gamma_k = \infty, \qquad \sum_{k=0}^{\infty} \gamma_k^2 < \infty$$

  then numerator finite but denominator $\to \infty$

- example: $\gamma_k = c/k$ for $c \in (0, \infty)$

Variations

- stochastic gradient methods
  - noisy unbiased subgradients
  - similar convergence result in expectation
- dual averaging
  - accumulates subgradients
  - also includes a prox-step (if desired)
  - has improved convergence compared to standard subgradient method

Gradient method

- assume f is closed convex and continuously differentiable
- optimality condition:

$$x = \operatorname*{argmin}_x f(x) \quad \Leftrightarrow \quad 0 = \nabla f(x) \quad \Leftrightarrow \quad x = x - \gamma \nabla f(x)$$

- the gradient method is given by

$$x^{k+1} = x^k - \gamma \nabla f(x^k)$$

- is it guaranteed to converge for some fixed γ?

Gradient method

- assume f is closed convex and continuously differentiable
- optimality condition:

$$x = \operatorname*{argmin}_x f(x) \quad \Leftrightarrow \quad 0 = \nabla f(x) \quad \Leftrightarrow \quad x = x - \gamma \nabla f(x)$$

- the gradient method is given by

$$x^{k+1} = x^k - \gamma \nabla f(x^k)$$

- is it guaranteed to converge for some fixed γ?
- no, not, e.g., for $f(x) = x^4$

Divergent example with fixed step-size

- $f(x) = x^4$, then gradient step is

$$x_{k+1} = x_k - \gamma 4 x_k^3 = x_k(1 - \gamma 4 x_k^2)$$

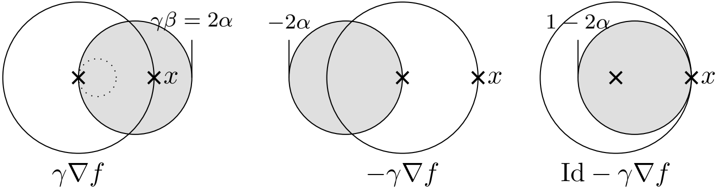
- let $x^0 > \frac{1}{2\sqrt{\gamma}}$, then $(1 - \gamma 4 x_0^2) < -1$ which implies that

$$x_1 < -x_0$$

- apply iteratively (with sign shift) to show divergence
- need also Lipschitz continuity of gradient!

Convergence

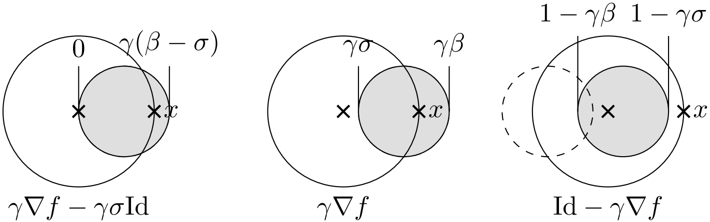
- assume that f is β-smooth
- equivalent to that ∇f is β-Lipschitz continuous ($\Rightarrow \frac{1}{\beta}$-cocoercive)
- assume that $\gamma = 2\alpha/\beta$ and $\alpha \in (0, 1)$ (i.e., $\gamma \in (0, \frac{2}{\beta})$)
- then $(\mathrm{Id} - \gamma\nabla f)$ is α-averaged



- iteration of α-averaged operator converges to fixed-point
- the convergence is sublinear

Stronger convergence

- assume that f is σ-strongly convex and β-smooth
- then $\gamma f - \frac{\gamma\sigma}{2}\|\cdot\|^2$ is $\gamma(\beta-\sigma)$-smooth
- and $\gamma(\nabla f - \sigma \mathrm{Id})$ is $\gamma(\beta-\sigma)$-Lipschitz
- then $(\mathrm{Id} - \gamma\nabla f)$ is $\max(\gamma\beta - 1, 1-\gamma\sigma)$-contractive



- here, we get linear convergence

Computing the step-size

- need step-size $\gamma \in (0, \frac{2}{\beta})$ to guarantee convergence
- need cocoercivity parameter $\frac{1}{\beta}$ to find convergent γ

Minimizing a quadratic approximation

- consider:

$$x^{k+1} = \operatorname*{argmin}_x \{ f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \tfrac{1}{2\gamma} \|x - x^k\|^2 \}$$

- optimality condition is $0 = \nabla f(x^k) + \frac{1}{\gamma}(x - x^k)$, i.e.,

$$x^{k+1} = x^k - \gamma \nabla f(x^k)$$

- so the gradient method minimizes a quadratic approximation of f
- since f is β-smooth, we have for all x, y:

$$f(y) \le f(x) + \langle \nabla f(x), y - x \rangle + \tfrac{\beta}{2} \|x - y\|^2$$

- if $\gamma \le \frac{1}{\beta}$, the quadratic approximation majorizes f
  (the gradient method is a majorization minimization algorithm)

Descent method

- the gradient method can be interpreted as a descent method
- since f is β-smooth, we have

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\beta}{2}\|x - y\|^2$$

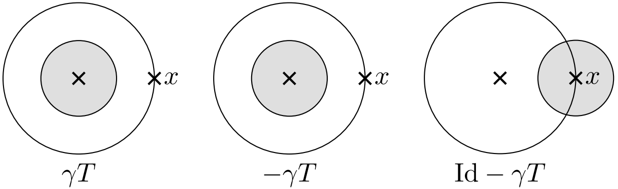
- let $y = x^{k+1} = x^k - \gamma \nabla f(x^k)$ and $x = x^k$, then

$$\begin{aligned} f(x^{k+1}) &\leq f(x^k) - \langle \nabla f(x^k), \gamma \nabla f(x^k) \rangle + \frac{\beta}{2}\|\gamma \nabla f(x^k)\|^2 \\ &\leq f(x^k) - \gamma(1 - \frac{\gamma\beta}{2})\|\nabla f(x^k)\|^2 \end{aligned}$$

- that is, it is a descent method if $\gamma(1 - \frac{\gamma\beta}{2}) > 0$ or $\gamma \in (0, \frac{2}{\beta})$ (same condition as before)

General Lipschitz operators

- suppose that T is β-Lipschitz:



- cannot make $\mathrm{Id} - \gamma T$ nonexpansive independent of γ
- iterating forward step of Lipschitz T not guaranteed to converge
- in convex function case Lipschitz $\Rightarrow$ cocoercivity
- cocoercivity is important property for convergence!
- if T cocoercive, we get convergence as for gradient method! (forward step method)

Accelerated versions

- here convergence means convergence in function value
- optimal convergence using gradient information is $O(1/k^2)$
- standard gradient method has nonoptimal convergence: $O(1/k)$
- accelerated scheme exists that achieves optimal rate $O(1/k^2)$
- it adds a very specific varying momentum term to iterates
- above holds in general sublinear case
- for linearly convergent case, similar acceleration can be made

Proximal point algorithm

- suppose that f is proper closed and convex and not differentiable
- optimality condition:

$$\begin{aligned} 0 \in \partial f(x) &\Leftrightarrow x \in x + \gamma \partial f(x) \\ &\Leftrightarrow x \in (\mathrm{Id} + \gamma \partial f)x \\ &\Leftrightarrow x = (\mathrm{Id} + \gamma \partial f)^{-1} x \\ &\Leftrightarrow x = \mathsf{prox}_{\gamma f}(x) \end{aligned}$$

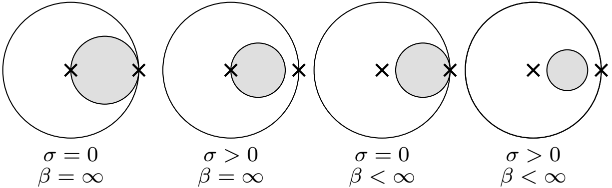
- iterate this to get proximal point algorithm

$$x^{k+1} = \mathsf{prox}_{\gamma f}(x^k) := \min_x \{ f(x) + \tfrac{1}{2\gamma} \|x - x^k\|^2 \}$$

Prox operator properties

recall prox operator properties (and $0 \leq \sigma \leq \beta$):

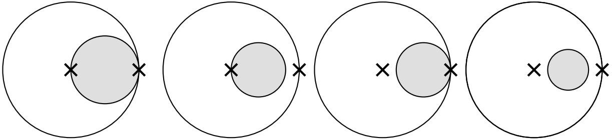
- f convex $\Rightarrow$ prox_f is 1-cocoercive or $\frac{1}{2}$-averaged
- f is σ-strongly convex $\Rightarrow$ prox_f is $(1+\sigma)$-cocoercive
- f is β-smooth $\Rightarrow$ prox_f is $\frac{\beta}{2(1+\beta)}$-averaged ($< \frac{1}{2}$-averaged)
- f is σ-strongly convex and β-smooth
  - $\beta > \sigma$: $\text{prox}_f - \frac{1}{1+\beta}\mathrm{Id}$ is $\frac{1}{\frac{1}{1+\sigma} - \frac{1}{1+\beta}}$-cocoercive
  - $\beta = \sigma$: $\text{prox}_f - \frac{1}{1+\beta}\mathrm{Id}$ is 0-Lipschitz



- all these are 1-cocoercive, hence $\frac{1}{2}$-averaged

Convergence

- since always $\frac{1}{2}$-averaged $\Rightarrow$ sublinear convergence
- if $\sigma > 0$ then prox_f is contractive $\Rightarrow$ linear convergence



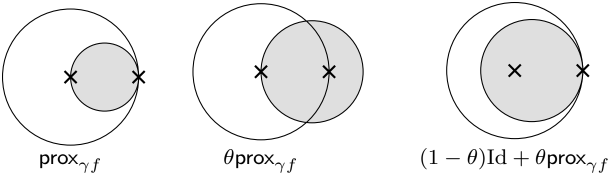
$\sigma = 0$, $\beta = \infty$ | $\sigma > 0$, $\beta = \infty$ | $\sigma = 0$, $\beta < \infty$ | $\sigma > 0$, $\beta < \infty$

Relaxed iterations

- we can relax the proximal point algorithm with $\theta \in (0, 2)$:

$$x^{k+1} = ((1-\theta)\mathrm{Id} + \theta \mathsf{prox}_{\gamma f})x^k$$

- example with $\theta = 1.5$:



- $\theta = 1.5$ gives $\alpha = 0.75$-averaged iteration
- $\theta > 1$ is called over-relaxation and $\theta < 1$ is called under-relaxation

Relation to averaged iteration

- let $\alpha = \frac{\theta}{2} \in (0, 1)$
- we can write the relaxed proximal point algorithm as

$$\begin{aligned} x^{k+1} &= ((1-\theta)\mathrm{Id} + \theta \mathsf{prox}_{\gamma f})x^k \\ &= ((1-2\alpha)\mathrm{Id} + 2\alpha \mathsf{prox}_{\gamma f})x^k \\ &= ((1-\alpha)\mathrm{Id} + \alpha(2\mathsf{prox}_{\gamma f} - \mathrm{Id}))x^k \\ &= ((1-\alpha)\mathrm{Id} + \alpha R_{\gamma f})x^k \end{aligned}$$

  where $R_{\gamma f} = 2\mathsf{prox}_{\gamma f} - \mathrm{Id}$ is the reflected resolvent
- since $R_{\gamma f}$ is nonexpansive, it is $\alpha = \frac{\theta}{2}$-averaged iteration

Iteration cost

- the problem to be solved is

$$\text{minimize} \quad f(x)$$

- the algorithm solves in each iteration

$$\min_x \{f(x) + \tfrac{1}{2\gamma}\|x - x^k\|^2\}$$

- often as difficult to solve as original problem
- (however, has nice convergence guarantees)

Resolvent method

- suppose A is maximally monotone
- we want to find x such that $0 \in Ax$
- condition:

$$\begin{aligned} 0 \in Ax &\Leftrightarrow x \in x + \gamma Ax \\ &\Leftrightarrow x \in (\mathrm{Id} + \gamma A)x \\ &\Leftrightarrow x = (\mathrm{Id} + \gamma A)^{-1}x \\ &\Leftrightarrow x = J_{\gamma A}x \end{aligned}$$

- construct an algorithm from this

$$x^{k+1} = J_{\gamma A}x^k$$

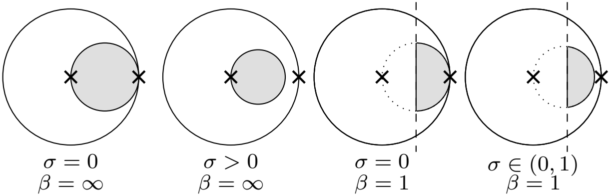
- if fixed-point found, inclusion problem solved
- if $A = \partial f$, we get proximal point algorithm
- (the resolvent method is also often called proximal point method)

Resolvent properties

recall prox operator properties (and $0 \leq \sigma \leq \beta$):

- A monotone $\Rightarrow$ J_A is 1-cocoercive or $\frac{1}{2}$-averaged
- A is σ-monotone $\Rightarrow$ J_A is $(1+\sigma)$-cocoercive
- A is β-Lipschitz $\Rightarrow$

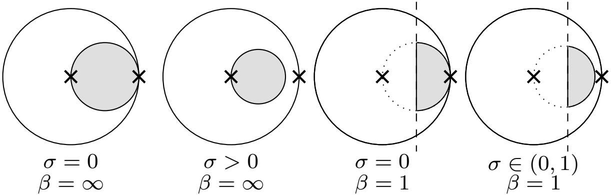
$$2\langle J_A x - J_A y, x - y\rangle \geq \|x-y\|^2 + (1-\beta^2)\|J_A x - J_A y\|^2$$



- all these are 1-cocoercive, hence $\frac{1}{2}$-averaged

Convergence

- since always $\frac{1}{2}$-averaged $\Rightarrow$ sublinear convergence
- if $\sigma > 0 \Rightarrow J_A$ contractive $\Rightarrow$ linear convergence

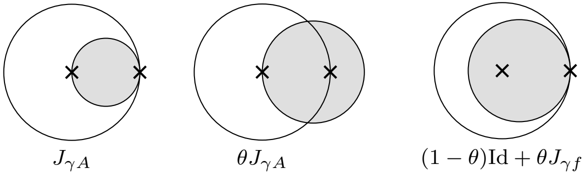


Relaxed iterations

- as with proximal point algorithm, we can relax with $\theta \in (0, 2)$:

$$x^{k+1} = ((1-\theta)\mathrm{Id} + \theta J_{\gamma A})x^k$$

- example with $\theta = 1.5$:



- equivalent to (as in proximal point case)

$$x^{k+1} = ((1-\alpha)\mathrm{Id} + \alpha R_{\gamma A})x^k$$

where $\alpha = \frac{\theta}{2}$ and $R_{\gamma A} = 2J_{\gamma A} - \mathrm{Id}$

Rewriting the iterates

- the iterations of the resolvent algorithm satisfies

$$
\begin{aligned}
& & x^{k+1} = (\mathrm{Id} + \gamma A)^{-1} x^k \\
\Leftrightarrow & & x^k \in (\mathrm{Id} + \gamma A) x^{k+1} \\
\Leftrightarrow & & 0 \in \gamma A x^{k+1} + (x^{k+1} - x^k)
\end{aligned}
$$

- iterates that satisfy this correspond to iteration of a $\frac{1}{2}$-averaged operator $J_{\gamma A}$

Resolvent method with skewed metric

- what if we have iterates that satisfy

$$0 \in \gamma A x^{k+1} + G(x^{k+1} - x^k)$$

  for some positive semi-definite G?
- assume that x^{k+1} is unique (holds, e.g., if G is positive definite)
- then $Gx^k \in (A+G)x^{k+1}$ and $x^{k+1} = (A+G)^{-1}Gx^k$
- let $T = (A+G)^{-1}G$, then T is $\frac{1}{2}$-averaged in G-norm

Proof of averagedness

- recall $T = (A+G)^{-1}G$, let $x^+ = Tx$, then

$$(A+G)x^+ = ATx + GTx \ni Gx$$

- then, we can choose $\bar{x}_1 \in ATx_1$ and $\bar{x}_2 \in ATx_2$ such that

$$\bar{x}_1 + GTx_1 = Gx_1, \qquad \bar{x}_2 + GTx_2 = Gx_2$$

- since A is monotone, we have

$$\langle \bar{x}_1 - \bar{x}_2, Tx_1 - Tx_2 \rangle \geq 0$$

- therefore

$$\begin{aligned}
\|Tx_1 - Tx_2\|_G^2 + 0 &\leq \langle G(Tx_1 - Tx_2), Tx_1 - Tx_2 \rangle \\
&\quad + \langle \bar{x}_1 - \bar{x}_2, Tx_1 - Tx_2 \rangle \\
&= \langle G(x_1 - x_2), Tx_1 - Tx_2 \rangle \\
&= \langle Tx_1 - Tx_2, x_1 - x_2 \rangle_G
\end{aligned}$$

- that is 1-cocoercive, $\frac{1}{2}$-averaged, firmly nonexpansive in G-norm

Convergence

- analyze convergence of $x^{k+1} = (A + G)^1 G x^k = T x^k$
- completion of squares gives

$$\begin{aligned}\|Tx_1 - Tx_2\|_G^2 &\le \langle Tx_1 - Tx_2, x_1 - x_2\rangle_G \\ &= \tfrac{1}{2}\|Tx_1 - Tx_2\|_G^2 + \tfrac{1}{2}\|x_1 - x_2\|_G^2 \\ &\quad - \tfrac{1}{2}\|(\mathrm{Id} - T)x_1 + (\mathrm{Id} - T)x_2\|_G^2\end{aligned}$$

- that is (compare to $\frac{1}{2}$-averaged, then $G = \mathrm{Id}$)

$$\|(\mathrm{Id} - T)x_1 + (\mathrm{Id} - T)x_2\|_G^2 \le \|x_1 - x_2\|_G^2 - \|Tx_1 - Tx_2\|_G^2$$

- as in normal $\frac{1}{2}$-averaged case, let $x_1 = x^k$, $x_2 = x^*$ where $Tx^* = x^*$:

$$\|(\mathrm{Id} - T)x^k\|_G^2 \le \|x^k - x^*\|_G^2 - \|Tx^k - Tx^*\|_G^2$$

or

$$\|x^k - x^{k+1}\|_G^2 \le \|x^k - x^*\|_G^2 - \|x^{k+1} - x^*\|_G^2$$

- telescope summation gives convergence in G-norm

Is resolvent algorithm useful?

- many algorithms can be seen as resolvent method for some maximally monotone operator A
- actually T is $\frac{1}{2}$-averaged with dom$T = \mathbb{R}^n \Leftrightarrow T = (\mathrm{Id} + A)^{-1}$ with A maximally monotone
- all algorithm that iterate $\frac{1}{2}$-averaged operators are resolvent algorithms
- if iterating averaged operator with other α, can be seen as resolvent method with under- or over-relaxation

Forward-backward splitting

- suppose that A is maximally monotone and B is $\frac{1}{\beta}$-cocoercive
- we want to find x such that

$$0 \in Ax + Bx$$

- for any $\gamma \in (0, \infty)$, such an x satisfies

$$\begin{aligned} 0 \in Ax + Bx &\iff -\gamma Bx \in \gamma Ax \\ &\iff (\mathrm{Id} - \gamma B)x \in (\mathrm{Id} + \gamma A)x \\ &\iff J_{\gamma A}(\mathrm{Id} - \gamma B)x = x \end{aligned}$$

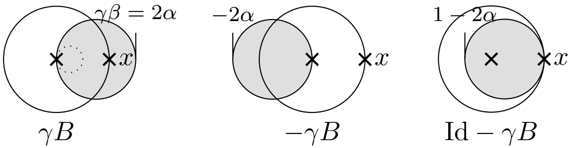
- construct algorithm from this

$$x^{k+1} = J_{\gamma A}(\mathrm{Id} - \gamma B)x^k$$

(first take forward step then backward (resolvent) step)

Convergence

- let $\gamma = 2\alpha/\beta$ and $\alpha \in (0,1)$
- then $(\mathrm{Id} - \gamma B)$ is α-averaged since B is $\frac{1}{\beta}$-cocoercive



- A is maximally monotone $\Rightarrow J_{\gamma A}$ is $\frac{1}{2}$-averaged (for any $\gamma > 0$)
- therefore, $J_{\gamma A}(\mathrm{Id} - \gamma B)$ is composition of averaged operators
- composition of averaged operators is averaged

  $\Rightarrow$ algorithm is iteration of averaged operator!

  $\Rightarrow$ sublinear convergence

Stronger convergence results

- A is σ-strongly monotone $\Rightarrow$ $J_{\gamma A}$ contractive
- B is σ-strongly monotone $\Rightarrow$ $(\mathrm{Id} - \gamma B)$ contractive (for appr. γ)
- in either of these cases $J_{\gamma A}(\mathrm{Id} - \gamma B)$ is contractive (composition of nonexp. and contractive operator is contractive)

  $\Rightarrow$ algorithm converges linearly
- (obviously, the contractions factors can be quantified)

Application to optimization

- suppose that f is β-smooth and g is proper closed convex
- we want to solve

$$\text{minimize} \quad f(x) + g(x)$$

- under suitable constraint qualification, it is equivalent to finding x

$$0 \in \nabla f(x) + \partial g(x)$$

- can apply FB splitting since ∇f cocoercive and ∂g monotone
- also called (primal) proximal gradient method

Projected gradient method

- assume that C is a nonempty closed and convex set
- let $g = \iota_C$ then FB-splitting or proximal gradient method becomes

$$\begin{aligned} x^{k+1} &= J_{\gamma g}(\mathrm{Id} - \gamma \nabla f)x^k \\ &= \mathsf{prox}_{\gamma g}(\mathrm{Id} - \gamma \nabla f)x^k \\ &= \mathsf{proj}_C(\mathrm{Id} - \gamma \nabla f)x^k \end{aligned}$$

since

$$\mathsf{prox}_{\gamma g} = \operatorname*{argmin}_x \{\iota_C(x) + \tfrac{1}{2\gamma}\|x - z\|^2\} = \operatorname*{argmin}_{x \in C} \|x - z\| =: \mathsf{proj}_C(z)$$

- that is, it is the projected gradient method
- proximal gradient method generalization of this

Convergence

- sublinear convergence in general case
- linear convergence under strong convexity assumptions on f or g
- (this follows from general analysis above)

Problem with composition

- assume f is β-smooth, g proper closed convex, L linear
- what if we want to solve

$$\text{minimize } f(x) + (g \circ L)(x) = f(x) + g(Lx)$$

- apply forward-backward splitting:

$$x^{k+1} = \text{prox}_{\gamma(g \circ L)}(\mathrm{Id} - \gamma \nabla f)x^k$$

- often $\text{prox}_{\gamma(g \circ L)}(z)$ expensive to compute:

$$\text{prox}_{\gamma(g \circ L)}(z) = \underset{x}{\operatorname{argmin}}(g(Lx) + \tfrac{1}{2\gamma}\|x - z\|^2\}$$

if $g(y) = \sum_i^m g_i(y_i)$, separability of prox lost due to L

Problem with composition

- we want again to solve

$$\text{minimize } f(x) + (g \circ L)(x) = f(x) + g(Lx)$$

- now with f being σ-strongly convex
- formulate dual problem

$$\text{minimize } (f^* \circ (-L^*))(\mu) + g^*(\mu) = f^*(-L^*\mu) + g^*(\mu)$$

- apply forward-backward splitting on dual:

$$\begin{aligned} \mu^{k+1} &= \text{prox}_{\gamma g^*}(\mathrm{Id} - \gamma\nabla(f^* \circ (-L^*)))\mu^k \\ &= \text{prox}_{\gamma g^*}(\mu^k + \gamma L \nabla f^*(-L^*\mu^k)) \end{aligned}$$

- operator L only gives rise to multiplication with L and L^*

Convergence

- dual problem

$$\text{minimize } (f^* \circ (-L^*))(\mu) + g^*(\mu)$$

- f is σ-strongly convex $\Rightarrow$
  - f^* is $\frac{1}{\sigma}$-smooth
  - $(f^* \circ (-L^*))$ is $\frac{\|L^*\|^2}{\sigma}$-smooth
  - $\nabla(f^* \circ (-L^*))$ is $\frac{\sigma}{\|L^*\|^2}$-cocoercive
- g^* proper closed convex
- therefore assumptions to apply FB-splitting on dual are met!

  $\Rightarrow$ sublinear convergence if $\gamma = 2\alpha\sigma/\|L^*\|^2$ and $\alpha \in (0, 1)$

Stronger convergence

- dual proximal gradient method (dual FB splitting)

$$\mu^{k+1} = \mathsf{prox}_{\gamma g^*}(\mathrm{Id} - \gamma \nabla (f^* \circ (-L^*)))\mu^k$$

- we get linear convergence if either operator is contractive
  - $\mathsf{prox}_{\gamma g^*}$ contractive if g^* is strongly convex iff g is smooth
  - $(\mathrm{Id} - \gamma \nabla (f^* \circ (-L^*)))$ contractive if $f^* \circ (-L^*)$ strongly convex (holds if f is smooth and L is surjective (has full row rank))

Solving the primal

- algorithm solves dual, can we find primal solution?
- rewrite algorithm

$$\mu^{k+1} = \mathsf{prox}_{\gamma g^*}(\mathrm{Id} + \gamma L \nabla f^*(-L^* \mu))\mu^k$$

by letting $x^k = \nabla f^*(-L^* \mu^k)$ to get

$$x^k = \nabla f^*(-L^* \mu^k)$$
$$\mu^{k+1} = \mathsf{prox}_{\gamma g^*}(\mu^k + \gamma L x^k)$$

Solving the primal cont'd

- we know that μ^k converges to fixed-point $\bar{\mu} \Rightarrow x^k \to \bar{x}$:

$$\bar{x} = \nabla f^*(-L^*\bar{\mu})$$
$$\bar{\mu} = \text{prox}_{\gamma g^*}(\bar{\mu} + \gamma L\bar{x})$$

- apply Fermat's rule to prox expression:

$$0 \in \partial g^*(\bar{\mu}) + \gamma^{-1}(\bar{\mu} - (\bar{\mu} + \gamma L\bar{x}) = \partial g^*(\bar{\mu}) - L\bar{x}$$

- recall that

$$x \in \partial f^*(-L^*\mu), \qquad\qquad Lx \in \partial g^*(\mu)$$

  are necessary and sufficient optimality conditions
- therefore, algorithm can output primal and dual optimal points

Reformulation

- consider Moreau's identity

$$\mathrm{prox}_{\gamma g^*}(\gamma z) = \gamma(z - \mathrm{prox}_{\gamma^{-1} g}(z))$$

- using this, the dual FB algorithm

$$\begin{aligned} x^k &= \nabla f^*(-L^* \mu^k) \\ \mu^{k+1} &= \mathrm{prox}_{\gamma g^*}(\mu^k + \gamma L x^k) \end{aligned}$$

can be written as

$$\begin{aligned} x^k &= \nabla f^*(-L^* \mu^k) \\ y^k &= \mathrm{prox}_{\gamma^{-1} g}(\gamma^{-1} \mu^k + L x^k) \\ \mu^{k+1} &= \mu^k + \gamma(L x^k - y^k) \end{aligned}$$

(where z in Moreau's identity is $\gamma^{-1}\mu^k + Lx^k$)

Reformulation cont'd

- state explicitly the gradient of the conjugate f^*

$$\nabla f^*(-L^*\mu) = \underset{x}{\operatorname{argmax}}\{\langle -L^*\mu, x\rangle - f(x)\} = \underset{x}{\operatorname{argmin}}\{f(x) + \langle x, L^*\mu\rangle\}$$

- state explicitly $\text{prox}_{\gamma^{-1}g}$:

$$\text{prox}_{\gamma^{-1}g}(\gamma^{-1}\mu^k + Lx^k) = \underset{y}{\operatorname{argmin}}\{g(y) + \langle \mu^k, Lx^k - y\rangle + \tfrac{\gamma}{2}\|y - Lx^k\|^2\}$$

- then dual proximal gradient method can be written as

$$\begin{aligned} x^k &= \underset{x}{\operatorname{argmin}}\{f(x) + \langle x, L^*\mu\rangle\} \\ y^k &= \underset{y}{\operatorname{argmin}}\{g(y) + \langle \mu^k, Lx^k - y\rangle + \tfrac{\gamma}{2}\|y - Lx^k\|^2\} \\ \mu^{k+1} &= \mu^k + \gamma(Lx^k - y^k) \end{aligned}$$

Several g functions

- assume we want to solve

$$\begin{array}{ll} \text{minimize} & f(x) + \sum_{i=1}^{k} g_i(y_i) \\ \text{subject to} & L_i x = y_i \text{ for all } i = 1, \ldots, k \end{array}$$

- assume that f is strongly convex and g_i are proper closed convex
- introduce

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}, \qquad L = \begin{bmatrix} L_1 \\ \vdots \\ L_k \end{bmatrix}, \qquad g(y) = \sum_{i=1}^{k} g_i(y_i)$$

- then problem is

$$\begin{array}{ll} \text{minimize} & f(x) + \sum_{i=1}^{k} g(y) \\ \text{subject to} & Lx = y \end{array}$$

- can apply forward-backward splitting to dual
- will get k parallel prox on the g_i^*:s

Alternative formulation

- consider solving $\min_x\{f(x)+g(x)\}$ and let

$$x^{k+1}=\underset{x}{\operatorname{argmin}}\{f(x^k)+\langle\nabla f(x^k),x-x^k\rangle+\tfrac{1}{2\gamma}\|x-x^k\|^2+g(x)\}$$

- Fermat's rule implies

$$\begin{aligned}0&\in\nabla f(x^k)+\gamma^{-1}(x^{k+1}-x^k)+\partial g(x^{k+1})\\&=\partial g(x^{k+1})+\gamma^{-1}(x^{k+1}-(x^k-\gamma\nabla f(x^k)))\\&=\gamma\partial g(x^{k+1})+x^{k+1}-(x^k-\gamma\nabla f(x^k))\end{aligned}$$

which is Fermat's rule for

$$x^{k+1}=\mathsf{prox}_{\gamma g}(\mathrm{Id}-\gamma\nabla f)x^k$$

i.e., the proximal gradient method

- can be analyzed as a descent method

Generalized metric

- assume that L is positive definite
- consider solving $\min_x\{f(x)+g(x)\}$ and let

$$x^{k+1} = \operatorname*{argmin}_x\{f(x^k) + \langle \nabla f(x^k), x - x^k\rangle + \tfrac{1}{2}\|x - x^k\|_L^2 + g(x)\}$$

- algorithm converges if f 1-smooth w.r.t. $\|\cdot\|_L^2$, i.e., if for all x, y

$$f(y) \le f(x) + \langle \nabla f(x), y - x\rangle + \tfrac{1}{2}\|x - y\|_L^2$$

- might give better approximation of f in algorithm
  $\Rightarrow$ might improve performance
- if $L = \gamma^{-1} I$, we get standard method

Remarks

- can use back-tracking if feasible γ not known
- back-tracking can improve performance
- can also use acceleration similarly to in the gradient method
- acceleration achieves optimal convergence rate
- acceleration methods are sensitive to errors in computations (reason: the momentum term keeps all old iterates)