Fixed-Point Interpretations of Large-Scale Convex Optimization Algorithms

Pontus Giselsson

Outline – Large-scale optimization methods

- Examples of large-scale problems
- Algorithm building blocks
- Problem formulation
- A unified fixed-point interpretation view
  - Forward-backward and Douglas-Rachford fixed-point mappings
  - Necessary and sufficient conditions for convergence
  - Some algorithm examples
  - A contraction factor result
- A tiny numerical example (time permitting)

Convex optimization applications

- Least squares
- Lasso, ridge regression, elastic net
- Support vector machines
- Logistic regression
- Sparse classification
- Matrix completion
- Model predictive control
- System identification
- Model reduction
- Portfolio optimization
- Signal reconstruction
- Trend filtering

Algorithm types and problem dimensions

Problem dimension	Algorithm type
small to medium scale (up to 1'000 variables)	Second-order methods (Newton's method, interior point)
large-scale (up to 100'000 variables)	First-order methods
huge-scale (more than 100'000 variables)	Stochastic, coordinate, parallel asynchronous first-order methods

In data rich fields, problems usually large to huge scale

Large-and huge scale algorithms

Will present unified view of:

- Projected gradient methods
- Proximal gradient methods
- Forward-backward splitting
- Douglas-Rachford splitting
- The alternating direction method of multipliers
- SAGA
- Finito/MISO
- SVRG
- Block-coordinate (proximal) gradient descent
- Block-coordinate consensus optimization
- (Three operator splitting methods)
- (Chambolle-Pock and Primal-dual methods)

First-order method building blocks

- (Sub-)gradients:

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

- Projections onto a sets C:

$$\Pi_C(z) = \operatorname*{argmin}_x (\|x - z\|_2 : x \in C)$$

- Proximal operators:

$$\operatorname{prox}_{\gamma g}(z) = \operatorname*{argmin}_x (g(x) + \tfrac{1}{2\gamma}\|x - z\|_2^2)$$

where $\gamma > 0$ is a parameter.

Prox is generalization of projection

- Introduce the indicator function of a set C

$$\iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{otherwise} \end{cases}$$

(this is an extended valued function, i.e., $\mathrm{dom}\iota_C = \mathbb{R} \cup \{\infty\}$)

- Then

$$\begin{aligned} \Pi_C(z) &= \underset{x}{\mathrm{argmin}}(\|x - z\|_2 : x \in C) \\ &= \underset{x}{\mathrm{argmin}}(\tfrac{1}{2}\|x - z\|_2^2 : x \in C) \\ &= \underset{x}{\mathrm{argmin}}(\tfrac{1}{2}\|x - z\|_2^2 + \iota_C(x)) \\ &= \mathrm{prox}_{\iota_C}(z) \end{aligned}$$

(projection onto C equals prox of indicator function of C)

Examples of proximal operators

- Quadratic function, $g(x) = \frac{1}{2}x^T Hx + h^T x$:

$$\mathrm{prox}_{\gamma g}(z) = (I + \gamma H)^{-1}(z - \gamma h)$$

- The squared 2-norm, $g(x) = \frac{1}{2}\|x\|_2^2$:

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

- The 2-norm, $g(x) = \|x\|_2$:

$$\mathrm{prox}_{\gamma g}(z) = \begin{cases} (1 - \gamma/\|z\|_2)z & \text{if } \|z\|_2 \geq \gamma \\ 0 & \text{otherwise} \end{cases}$$

- Affine subspace, $V = \{x : Ax = b\}$:

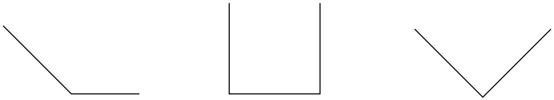
$$\mathrm{prox}_{\iota_V}(z) = \Pi_V(z) = z - A^T(AA^T)^{-1}(Az - b)$$

Piece-wise linear function

- Define $h_i \;:\; \mathbb{R} \to \overline{\mathbb{R}}$ is

$$h_i(x) = \begin{cases} c_l(l - x) & \text{if } x \le l \\ 0 & \text{if } l \le x \le u \\ c_u(x - u) & \text{if } x \ge u \end{cases}$$

  where $c_l, c_u \in (0, \infty]$ (∞ included) and $l \le u$
- graphical representations of different h_i



$c_l = 1,\ c_u = 0$ $\quad$ $l = 1,\ u = \infty$

$c_l = c_u = \infty$ $\quad$ $l = -1,\ u = 1$

$c_l = c_u = 1$ $\quad$ $l = u = 0$

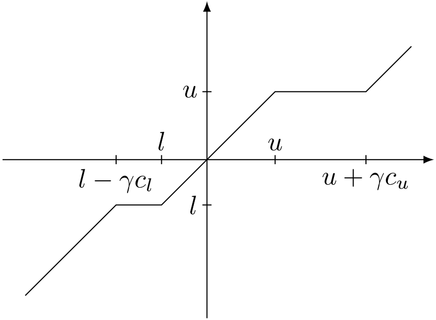
- special cases of h_i
  - hinge loss (SVM)
  - upper and lower bounds
  - "soft" upper and lower bounds
  - absolute value

Prox of h_i

- Prox of h_i:

$$\mathrm{prox}_{\gamma h_i}(z) = \begin{cases} z + \gamma c_l & \text{if } z \leq l - \gamma c_l \\ l & \text{if } l - \gamma c_l \leq z \leq l \\ z & \text{if } l \leq z \leq u \\ u & \text{if } u \leq z \leq u + \gamma c_u \\ z - \gamma c_u & \text{if } z \geq u + \gamma c_u \end{cases}$$

- Graphical representation ($l = -1, u = 1.5, \gamma c_l = 1, \gamma c_u = 2$):



Examples prox h_i

- Hinge loss, $g = h_i$ with $l = 1$, $u = \infty$, $c_l = 1$, $c_u = 0$:

$$\mathrm{prox}_{\gamma g}(z) = \begin{cases} z + \gamma & \text{if } z \leq 1 - \gamma \\ 1 & \text{if } 1 - \gamma \leq z \leq 1 \\ z & \text{if } z \geq 1 \end{cases}$$

- Absolute value, $g = h_i$ with $l = u = 0$ and $c_l = c_u = 1$:

$$\mathrm{prox}_{\gamma g}(z) = \begin{cases} z + \gamma & \text{if } z \leq -\gamma \\ 0 & \text{if } -\gamma \leq z \leq \gamma \\ z - \gamma & \text{if } z \geq \gamma \end{cases}$$

- Upper and lower bounds, $g = h_i$ with $l < u$ and $c_l = c_u = \infty$:

$$\mathrm{prox}_{\gamma g}(z) = \begin{cases} l & \text{if } z \leq l \\ z & \text{if } l \leq z \leq u \\ u & \text{if } u \leq z \end{cases}$$

Prox of separable functions

- Separable functions $g(x) = \sum_{i=1}^{n} g_i(x_i)$ where $x = (x_1, \ldots, x_n)$:

$$\operatorname{prox}_{\gamma g}(z) = \begin{pmatrix} \operatorname{prox}_{\gamma g_1}(z_1) \\ \vdots \\ \operatorname{prox}_{\gamma g_n}(z_n) \end{pmatrix}$$

- Decomposes into n individual proxes
- 1-norm $\|x\|_1$, upper/lower bounds, hinge loss constructed from h_i

Some prox/gradient computational costs

$g : \mathbb{R}^n \to \mathbb{R}^n$	prox cost	grad cost	comment
$\frac{1}{2}x^T Hx + h^T x$	$O(n^3)$	$O(n^2)$	Sparse H cheaper prox $O(n^2)$ after factorization
$\frac{1}{2}\|Ax - b\|_2^2$	$O(m^2 n)$	$O(mn)$	$A \in \mathbb{R}^{m\times n}$, $m \leq n$ Sparse A cheaper prox $O(mn)$ after factorization
$\frac{1}{2}\|x\|^2$	$O(n)$	$O(n)$	
$\|x\|$	$O(n)$	–	
$\iota_{\{x:Ax=b\}}$	$O(m^2 n)$	–	$A \in \mathbb{R}^{m\times n}$, $m \leq n$ Sparse A cheaper
$\sum_{i=1}^n h_i(x_i)$	$O(n)$	–	1-norm, upper/lower bounds hinge loss
$\sum_i \log(1 + e^{-y_i x_i})$	??	$O(n)$	Logistic loss prox requires iterative method

These and more implemented in ProximalOperators package in Julia

Pre and post compositions

- Precomposition with $A \in \mathbb{R}^{m \times n}$: $g(x) = f(Ax)$
  - Gradient cost: $O(mn) + cost(\nabla f)$
  - Prox cost: typically higher than for f
- Postcomposition with A: $g(x) = \inf(f(y) : Ay = x)$
  - Not differentiable even if f is
  - Prox cost: typically higher than for f

Prox as resolvent

- The proximal operator satisfies

$$\operatorname{prox}_{\gamma g} = (I + \gamma \partial g)^{-1}$$

  where

  - ∂g is the subdifferential operator
  - $(\cdot)^{-1}$ is the inverse operator
  - $(I + \gamma \partial g)^{-1}$ is called the *resolvent*

- Reason: optimality condition for the prox-computation:

$$\begin{aligned}
& x = \operatorname{prox}_{\gamma g}(z) && \Leftrightarrow \\
& x = \underset{x}{\operatorname{argmin}}\{g(x) + \tfrac{1}{2\gamma}\|x - z\|^2\} && \Leftrightarrow \\
& 0 \in \gamma \partial g(x) + x - z && \Leftrightarrow \\
& z \in (I + \gamma \partial g)x && \Leftrightarrow \\
& x = (I + \gamma \partial g)^{-1} z
\end{aligned}$$

Outline – Large-scale optimization methods

- Examples of large-scale problems
- Algorithm building blocks
- **Problem formulation**
- A unified fixed-point interpretation view
  - Forward-backward and Douglas-Rachford fixed-point mappings
  - Necessary and sufficient conditions for convergence
  - Some algorithm examples
  - A contraction factor result
- A tiny numerical example (time permitting)

Problem formulations

- Most algorithms solve problems of the form

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

  where f, g may be extended-valued: $f, g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$
- Models e.g., constrained problems through

$$\text{minimize } f(x) + \iota_C(x)$$

  where ι_C is indicator function for set C

Consensus formulation

- What if we want to solve problems of the form

$$\text{minimize } \tfrac{1}{n}\sum_{i=1}^{n} f_i(x)$$

- One approach is to use consensus formulation:

$$\text{minimize } \underbrace{\tfrac{1}{n}\sum_{i=1}^{n} f_i(x_i)}_{f(\mathbf{x})} + \underbrace{\iota_C(x_1, \ldots, x_n)}_{g(\mathbf{x})}$$

with individual x_i for each f_i and a consensus constraint

$$C := \{(x_1, \ldots, x_n) : x_1 = \cdots = x_n\}$$

- Problem reduces to two function problem from before
- (Also called divide and concur)

Outline – Large-scale optimization methods

- Examples of large-scale problems
- Algorithm building blocks
- Problem formulation
- **A unified fixed-point interpretation view**
    - Forward-backward and Douglas-Rachford fixed-point mappings
    - Necessary and sufficient conditions for convergence
    - Some algorithm examples
    - A contraction factor result
- A tiny numerical example (time permitting)

Algorithms – An abstract view

- Most algorithms translate problem to fixed-point problem:

$$\text{find } x^\star \text{ such that } Tx^\star = x^\star$$

  where T is referred to as fixed-point operator (mapping)
- Fixed-points of T have close relationship to solution of problem
- Most algorithms are based on one of the following:
  - The forward-backward map
  - The Douglas-Rachford map

The forward-backward map

- Assume ∇f is Lipschitz and f is convex, g is convex, then (CQ)

$$\begin{aligned} x \in \operatorname{argmin}\{f(x) + g(x)\} &\Leftrightarrow 0 \in \nabla f(x) + \partial g(x) \\ &\Leftrightarrow -\gamma \nabla f(x) \in \nabla + \gamma \partial g(x) \\ &\Leftrightarrow (I - \gamma \nabla f)x \in (I + \gamma \partial g)x \\ &\Leftrightarrow (I + \gamma \partial g)^{-1}(I - \gamma \nabla f)x \in x \\ &\Leftrightarrow \operatorname{prox}_{\gamma g}(I - \gamma \nabla f)x = x \end{aligned}$$

- The map $\operatorname{prox}_{\gamma g}(I - \gamma \nabla f)$ is the FB map
- Its fixed-points coincide with solutions to optimization problem
- Reverse order gives backward-forward operator $(I - \gamma \nabla f)\operatorname{prox}_{\gamma g}$:

$$\operatorname{Argmin}\{f(x) + g(x)\} = \operatorname{prox}_{\gamma g}\left(\operatorname{Fix}\left((I - \gamma \nabla f)\operatorname{prox}_{\gamma g}\right)\right)$$

where $\operatorname{Fix} T = \{x : x = Tx\}$

The Douglas-Rachford map

- Let $R_{\gamma f} = 2\mathrm{prox}_{\gamma f} - I$ be the *reflector* or *reflected resolvent*
- It can be shown that

$$\operatorname*{Argmin}_x\{f(x) + g(x)\} = \mathrm{prox}_{\gamma g}(\mathrm{Fix} R_{\gamma f} R_{\gamma g})$$

- The composition of reflected resolvents $R_{\gamma f} R_{\gamma g}$ is DR map
- Fixed-point solves optimization problem after prox-step

Why these mappings?

- They have the favorable property of being nonexpansive
- Forward-backward operator
  - Assume f, g convex, ∇f L-Lipschitz, and $\gamma \in (0, \frac{1}{2L})$
  - Then $\mathrm{prox}_\gamma(I - \gamma\nabla f)$ is nonexpansive
- Douglas-Rachford operator
  - Assume f, g convex and $\gamma \in (0, \infty)$
  - Then $R_{\gamma f} R_{\gamma g}$ is nonexpansive
- Reason, building blocks have similar favorable properties

Nonexpansive

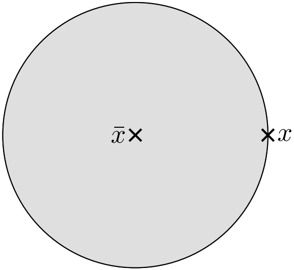
- The operators T are nonexpansive: for all x, y:

$$\|Tx - Ty\| \leq \|x - y\|$$

- Let $y = \bar{x}$ where $\bar{x} = T\bar{x}$ is a fixed-point to T, then

$$\|Tx - \bar{x}\| \leq \|x - \bar{x}\|$$

- 2D graphical representation



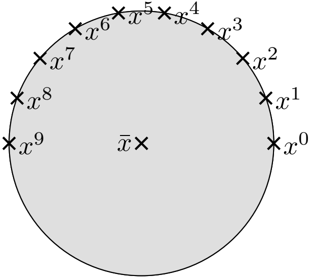
Tx in gray area (distance to fixed-point not increased)

Iterating T

- The iteration

$$x^{k+1} = Tx^k$$

  is not guaranteed to converge to a fixed-point
- Example: T is a rotation



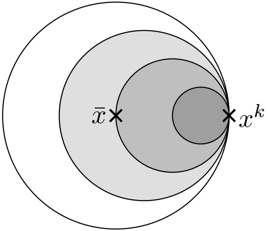
- Why is nonexpansiveness a useful property?

The role of α-averaging

- We consider averaged iteration of the nonexpansive mapping T:

$$x^{k+1} = (1 - \alpha)x^k + \alpha T x^k$$

  where $\alpha \in (0, 1)$
- 2D example on where x^{k+1} can end up for different α ($\bar{x} \in \mathrm{Fix}T$):



– $\alpha = 1$ – $\alpha = 0.75$ – $\alpha = 0.5$ – $\alpha = 0.25$

- Distance to fixed-points decreased if $\alpha \in (0, 1)$ and $Tx^k \neq x^k$

Property of α-averaged operator

- Let $S = (1-\alpha)I + \alpha T$ and $x^{k+1} = Sx^k$, then it can be shown

$$\|x^{k+1} - z\|^2 \leq \|x^k - z\|^2 - \beta\|x^k - Sx^k\|^2$$

  for all $z \in \mathrm{Fix}S = \mathrm{Fix}T$ and some $\beta > 0$
- $\|x^k - z\|^2$ is Lyapunov function and $\|x^k - Sx^k\|$ gives decrease
- Consequence:
  - $(\|x^k - z\|)_{k\geq 0}$ converges for all $z \in \mathrm{Fix}T$
  - $\|x^k - Sx^k\| = \alpha\|x^k - Tx^k\| \to 0$ as $k \to \infty$

  which is sufficient to show convergence towards a fixed-point

Many different ways to find fixed-point

- Many algorithms for large-scale optimization are of the form:

$$z^{k+1} := (1-\alpha)z^k + \alpha\hat{T}z^k = z^k - \alpha(z^k - \hat{T}z^k)$$

  where $\alpha \in (0,1)$ and $\hat{T}$ is either:
  - The full operator T (large-scale)
  - A randomized coordinate block update operator of T (huge-scale)
  - A stochastic approximation of T (huge-scale)
- The expected z^{k+1} given z^k for both stochastic methods satisfy:

$$\mathbb{E}_k z^{k+1} = z^k - \alpha(z^k - Tz^k)$$

  they are unbiased stochastic versions of the full operator method

Finding fixed-point of nonexpansive mapping

- The sufficient conditions:
  1. $(\|z - x^k\|)_{k \geq 0}$ converges for all $z \in \mathrm{Fix} T$
  2. $\|Tx^k - x^k\| \to 0$ as $k \to \infty$

  are also necessary conditions
- All orbits from algorithms that find fixed-point satisfy these

How to guarantee conditions – Deterministic case

- A Lyapunov function of the form

$$\|z^{k+1} - z^\star\|_2^2 + \kappa_{k+1} \leq \|z^k - z^\star\|_2^2 + \kappa_k - \gamma_k$$

  where $\gamma_k \geq 0$ and $\kappa_k \geq 0$ satisfy
  - $\gamma_k \to 0$ implies $\|Tx^k - x^k\| \to 0$
  - $\|Tx^k - x^k\| \to 0$ implies $\kappa_k \to 0$

- Easy to verify that

$$(\gamma_k)_{k\geq 0} \text{ is summable and } (\|z^k - z^\star\|^2 + \kappa_k)_{k\geq 0} \text{ converges}$$

  therefore $\|Tx^k - x^k\| \to 0$ and $\kappa_k \to 0$ which implies

$$\|Tx^k - x^k\| \to 0 \quad \text{and} \quad (\|z^k - z^\star\|^2)_{k\geq 0} \text{ converges}$$

  i.e., $z^k \to \mathrm{Fix} T$

How to guarantee conditions – Stochastic case

- A stochastic Lyapunov function of the form

$$\mathbb{E}_k\|z^{k+1} - z^\star\|_2^2 + \kappa_{k+1} \leq \|z^k - z^\star\|_2^2 + \kappa_k - \gamma_k$$

  where $\gamma_k \geq 0$ and $\kappa_k \geq 0$ as before
- The Robbins-Siegmund supermartingale theorem guarantees (a.s.)

$$(\gamma_k)_{k\geq 0} \text{ is summable and } (\|z^k - z^\star\|^2 + \kappa_k)_{k\geq 0} \text{ converges}$$

  which (by same arguments as before) implies $z^k \to \mathrm{Fix}T$ a.s.

Forward-backward splitting

- Solves: minimize $f(x) + g(x)$
- Applicable when f is Lipschitz differentiable
- Full update algorithm: Iterates forward-backward map:

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

  (i.e., uses $\alpha = 1$ since mapping already averaged)
- One forward (gradient) step and one backward (prox) step
- Special case: (projected) gradient method
- Converges to solution of optimization problem
- Cutting the algorithm differently gives backward-forward method:

$$z^{k+1} = (I - \gamma \nabla f)\mathrm{prox}_{\gamma g} z^k$$

Douglas-Rachford splitting

- Solves: minimize $f(x) + g(x)$
- Applicable when f and g are proper closed convex
- Averaged iteration of Douglas-Rachford map with $\alpha \in (0, 1)$:

$$z^{k+1} = (1 - \alpha)z^k + \alpha(2\mathrm{prox}_{\gamma f} - I)(2\mathrm{prox}_{\gamma g} - I)z^k$$

- Two backward (prox) steps
- Converges to fixed-point $z^\star$ of DR operator T
- Solution to optimization problem is $\mathrm{prox}_{\gamma g} z^\star$

Alternating direction method of multipliers (ADMM)

- Solves:

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

which is equivalent to solve: minimize $\hat{f}(z) + \hat{g}(z)$, where

$$\hat{f}(z) = \inf_x (f(x) : Ax = -z), \quad \hat{g}(z) = \inf_y (g(y) : By = c + z)$$

- Applicable when f and g are proper closed convex
- ADMM is averaged iteration of DR map applied to $\hat{f}$ and $\hat{g}$:

$$z^{k+1} = (1-\alpha)z^k + \alpha(2\mathrm{prox}_{\gamma\hat{f}} - I)(2\mathrm{prox}_{\gamma\hat{g}} - I)z^k$$

- Two backward (prox) steps on more complicated image functions
- If $A = I$, $B = -I$, and $c = 0$, ADMM=DR

ADMM cont'd

- Algorithm becomes (if subproblems can be solved):

$$
\begin{aligned}
x^k &:= \operatorname*{argmin}_x (f(x) + \tfrac{1}{2\gamma}\|Ax + z^k\|_2^2) \\
y^k &:= \operatorname*{argmin}_y (g(y) + \tfrac{1}{2\gamma}\|By + 2Ax^{k+1} + z^k - c\|_2^2) \\
z^{k+1} &:= z^k + 2\alpha(Ax^{k+1} + By^{k+1} - c)
\end{aligned}
$$

- It can also for $\alpha = \frac{1}{2}$ be implemented as

$$
\begin{aligned}
y^{k+1} &:= \operatorname*{argmin}_y (g(y) + \tfrac{1}{2\gamma}\|Ax^k + By - c + \gamma\lambda^k\|_2^2) \\
x^{k+1} &:= \operatorname*{argmin}_x (f(x) + \tfrac{1}{2\gamma}\|Ay + Bz^{k+1} - c + \gamma\lambda^k\|_2^2) \\
\lambda^{k+1} &:= \lambda^k + \gamma^{-1}(Ax^{k+1} + By^{k+1} - c)
\end{aligned}
$$

(this is standard formulation of ADMM)

Block-coordinate methods

- Decompose the T operator into m blocks:

$$T = \begin{pmatrix} (T)_1 \\ \vdots \\ (T)_m \end{pmatrix}$$

- Randomized block-coordinate update algorithm:
  1. Select an index $j \in \{1, \ldots, m\}$ at random with probabilities q_j
  2. Update block j according to:
  $$z_j^{k+1} := z_j^k - \tfrac{\alpha}{q_j}(z_j^k - (T)_j z^k)$$
  3. Leave the others: $z_i^{k+1} = z_i^k$ for all $i \neq j$
- Let $\widetilde{T}_j = (I_1, \ldots, (T_j), \ldots, I_m)$: Expected value of z^{k+1} given z^k:

$$\begin{aligned}\mathbb{E}_k z^{k+1} &= \mathbb{E}_k(z^k - \tfrac{\alpha}{q_j}(z^k - \widetilde{T}_j z^k)) = z^k - \sum_{j=1}^{m} \tfrac{q_j \alpha}{q_j}(z^k - \widetilde{T}_j z^k) \\ &= z^k - \alpha(z^k - T z^k)\end{aligned}$$

- Randomization is needed to guarantee convergence (a.s.)
- Efficient if m steps of block-method as expensive as one eval of T

Algorithm examples

- Forward-backward if g separable coordinate gradients of f cheap
- Consensus formulation and Douglas-Rachford
- Consensus formulation and backward-forward (Finito/MISO)

Stochastic backward-forward method

- Consider problems of the form

$$\text{minimize } \tfrac{1}{n}\sum_{i=1}^{n} f_i(x) + g(x)$$

  where f_i have L_i-Lipschitz gradients and g is prox-friendly
- Consider the backward-forward operator

$$T = (I - \gamma\nabla f)\mathrm{prox}_{\gamma g} = \tfrac{1}{n}\sum_{i=1}^{n}(I - \gamma\nabla f_i)\mathrm{prox}_{\gamma g}$$

- Algorithm with stochastic approximation of T:
  1. Randomly select an index $i \in \{1, \ldots, n\}$ with probability p_i
  2. Set: $z^{k+1} := z^k - \frac{\alpha_k}{np_i}(z^k - (I - \gamma\nabla f_i)\mathrm{prox}_{\gamma g} z^k)$
- Expected value of z^{k+1} given z^k with $T_i := (I - \gamma\nabla f_i)\mathrm{prox}_{\gamma g}$ is:

$$\mathbb{E}_k z^{k+1} = z^k - \mathbb{E}_k \tfrac{\alpha_k}{np_j}(z^k - T_i z^k) = z^k - \alpha_k(z^k - Tz^k)$$

- Advantage: Cheaper iterations than using ∇f if n large
- Drawback: Requires diminishing step-sizes to converge

Reduced variance stochastic methods

- Reduce variance by remembering old gradients $\nabla f_i x^{k-d_k}$
- An algorithm (SAGA):
  1. Select an index $i \in \{1, \ldots, n\}$ at random with probabilities p_i
  2. Update z and w-vectors according to:

$$z^{k+1} := w^k - \gamma \left(\tfrac{1}{np_i} \nabla f_i(w^k) - \tfrac{1}{np_i} y_i^k + \tfrac{1}{n} \sum_{j=1}^{n} y_j^k \right)$$
$$w^{k+1} := \mathrm{prox}_{\gamma g} z^{k+1}$$

  3. Update y-vectors according to:

$$y_i^{k+1} = \nabla f_i(w^k)$$

  4. Leave the others: $y_i^{k+1} = y_i^k$ for all $i \neq j$
- Expected value of z^{k+1} given z^k and y_i^k is:

$$\begin{aligned} \mathbb{E}_k z^{k+1} &= w^k - \tfrac{\gamma}{n} \sum_{j=1}^{n} y_j^k - \sum_{i=1}^{n} p_i \left(\tfrac{\gamma}{np_i} \nabla f_i(w^k) - \tfrac{\gamma}{np_i} y_i^k \right) \\ &= w^k - \gamma \nabla f(w^k) = (I - \gamma \nabla f) \mathrm{prox}_{\gamma g} z^k \end{aligned}$$

- Converges (a.s.) with fixed (but restricted) step-size
- Many variations in how and which y-vectors that are updated exist

RVSBC methods

- RVSBC - Reduced Variance Stochastic Block Coordinate
- Combine reduced variance and block coordinate methods

Caveats

- Algorithm parameters much be chosen to guarantee convergence!
- Stochastic methods should be implemented in real language
  In MATLAB, performance benefits not revealed due to for loops

A contraction factor result

- Recently many linear convergence results for ADMM when:
  - f strongly convex, differentiable, and ∇f Lipschitz
  - g convex (possibly extended-valued)
- Reason: The DR-map $R_{\gamma f} R_{\gamma g}$ becomes contractive

Example: LASSO

- Consider the LASSO problem:

$$\text{minimize} \quad \underbrace{\tfrac{1}{2}\|Ax-b\|_2^2}_{f(x)} + \underbrace{\lambda\|x\|_1}_{g(x)}$$

  with $A \in \mathbb{R}^{m\times n}$ and $m < n$
- f is differentiable with $\|A\|^2$-Lipschitz gradient
- 1-norm non-differentiable $\Rightarrow$ must use prox
- Can apply forward backward or Douglas-Rachford algorithm
- Possible (full operator) building blocks

$$\begin{aligned}
\nabla f(z) &= A^T(Az-b) && O(mn)\\
\mathrm{prox}_{\gamma f}(z) &= (I+\gamma A^TA)^{-1}(z+\gamma A^Tb) && O(nm^2)\ (O(nm))\\
\mathrm{prox}_{\gamma g}(z) &= \begin{cases} z+\gamma & \text{if } z \le -\gamma\\ 0 & \text{if } -\gamma \le z \le \gamma\\ z-\gamma & \text{if } z \ge \gamma \end{cases} && O(n)
\end{aligned}$$

LASSO: FB and DR

- Forward-backward algorithm ($O(mn)$ / iter):

$$\begin{aligned} z^k &:= x^k - \gamma A^T(Ax^k - b) \\ x^{k+1} &:= \operatorname{prox}_{\gamma\lambda\|\cdot\|_1}(z^k) \end{aligned}$$

- Douglas-Rachford algorithm ($O(m^2 n)$ first iter, then $O(mn)$):

$$\begin{aligned} x^k &:= (I + \gamma A^T A)^{-1}(z^k + \gamma A^T b) \\ y^k &:= \operatorname{prox}_{\gamma\lambda\|\cdot\|_1}(2x^k - z^k) \\ z^{k+1} &:= z^k + \alpha(y^k - x^k) \end{aligned}$$

LASSO: Coordinate update method

- The forward-backward operator has block-structure:

$$Tx^k = \begin{pmatrix} \mathrm{prox}_{\gamma\lambda|\cdot|}(x_1^k - a_1^T(Ax^k - b)) \\ \vdots \\ \mathrm{prox}_{\gamma\lambda|\cdot|}(x_n^k - a_n^T(Ax^k - b)) \end{pmatrix}$$

  where $a_i \in \mathbb{R}^m$ are columns of $A \in \mathbb{R}^{m\times n}$: $A = [a_1, a_2, \ldots, a_n]$
- Blocks seem expensive to evaluate due to $Ax^k - b$
- Due to linearity, can store and update $Ax^k - b$ according to

$$\begin{aligned} Ax^k - b &= (Ax^{k-1} - b) + A(x^k - x^{k-1}) \\ &= (Ax^{k-1} - b) + \underbrace{a_{j_{k-1}}(x_{j_{k-1}}^k - x_{j_{k-1}}^{k-1})}_{\text{cheap update}} \end{aligned}$$

  where last step holds since only $x_{j_{k-1}}$ updated
- Complexity per iteration: $O(m)$
- Can run roughly n iterations at cost of one FB iteration

LASSO: Reduced variance stochastic gradient

- To fit algorithm, write LASSO problem equivalently (divide by m):

$$\text{minimize} \quad \underbrace{\tfrac{1}{2m}\sum_{i=1}^{m}(a_i^T x - b_i)^2}_{f(x)} + \underbrace{\tfrac{\lambda}{m}\|x\|_1}_{g(x)}$$

where $a_i \in \mathbb{R}^n$ now are rows of $A \in \mathbb{R}^{m\times n}$: $A = [a_1^T, \ldots, a_n^T]^T$

- The gradient can be written as:

$$\nabla f(x) = \tfrac{1}{m}\sum_{i=1}^{m} a_i(a_i^T x - b_i)$$

- The backward-forward operator is

$$T = (I - \gamma\nabla f)\mathrm{prox}_{\gamma g} = \tfrac{1}{m}\sum_{i=1}^{m}(I - \tfrac{\gamma}{m}a_i(a_i^T(\cdot) - b_i))\mathrm{prox}_{\frac{\gamma\lambda}{m}\|\cdot\|_1}$$

LASSO: Reduced variance stochastic gradient

- SAGA algorithm:
  1. Select an index $i \in \{1, \ldots, n\}$ at random with probabilities p_i
  2. Update z and w-vectors according to:

$$
\begin{aligned}
z^{k+1} &:= w^k - \alpha \left(\tfrac{1}{np_i} a_i(a_i^T w^k - b_i) - \tfrac{1}{np_i} y_i^k + \tfrac{1}{n} \sum_{j=1}^{n} y_j^k \right) \\
w^{k+1} &:= \operatorname{prox}_{\alpha g} z^{k+1}
\end{aligned}
$$

  3. Update y-vectors according to:

$$
y_i^{k+1} = a_i(a_i^T w^k - b_i)
$$

  4. Leave the others: $y_i^{k+1} = y_i^k$ for all $i \neq j$
- In practice: Store $(a_i^T w^k - b_i) \in \mathbb{R}$ and multiply by a_i when used
- Complexity per iteration: $O(n)$
- Can run roughly m iterations of at same cost as one FB iteration

Numerical example

Randomly generated $A \in \mathbb{R}^{m \times n}$ with $m = 250$, $n = 300$:

Algorithm	FB	DR	CD	SAGA
Cost/iter	$O(mn)$	$O(m^2n), O(mn)$	$O(m)$	$O(n)$
Iters	542	107	33315	62848
Weighted iters	542	357	111	251

Randomly generated $A \in \mathbb{R}^{m \times n}$ with $m = 50$, $n = 300$:

Algorithm	FB	DR	CD	SAGA
Cost/iter	$O(mn)$	$O(m^2n), O(mn)$	$O(m)$	$O(n)$
Iters	1644	199	35376	77714
Weighted iters	1664	249	118	1554

Thank you

Questions?