Proximal Algorithms

Neal Parikh and Stephen Boyd

Stanford University

Based on 'Proximal Algorithms', *Foundations and Trends in Optimization*, 2014.

Outline

Proximal operator

- proximal operator of $f : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is

$$\mathbf{prox}_{\lambda f}(v) = \operatorname*{argmin}_x \left(f(x) + (1/2\lambda)\|x - v\|_2^2 \right)$$

  with parameter $\lambda > 0$

- f may be nonsmooth, have embedded constraints, ...
- evaluating $\mathbf{prox}_f$ involves solving a convex optimization problem
- can evaluate via standard methods like BFGS, but very often has an analytical solution or simple specialized linear-time algorithm

Generalized projection

- proximal operator of an indicator function of a convex set is projection:

$$\mathbf{prox}_{\lambda I_{\mathcal{C}}}(v) = \Pi_{\mathcal{C}}(v) = \operatorname*{argmin}_{x \in \mathcal{C}} \|x - v\|_2$$

- many properties carry over
- **example**: projection onto box $\mathcal{C} = \{x \mid l \preceq x \preceq u\}$ given by

$$(\Pi_{\mathcal{C}}(v))_k = \begin{cases} l_k & v_k \leq l_k \\ v_k & l_k \leq v_k \leq u_k \\ u_k & v_k \geq u_k \end{cases}$$

Quadratic functions

- if $f(x) = (1/2)x^T Px + q^T x + r$, then

$$\mathbf{prox}_{\lambda f}(v) = (I + \lambda P)^{-1}(v - \lambda q)$$

- if P is dense and direct method is used, costs $O(n^3)$ flops to factor and then $O(n^2)$ to backsolve on subsequent evaluations of $\mathbf{prox}_{\lambda f}$
- still get some discount if P is sparse or has some structure
- if using iterative method like CG/LSQR, warm start beginning at v

Separable sum

- if f is block separable, so $f(x) = \sum_{i=1}^N f_i(x_i)$, then

$$(\mathbf{prox}_f(v))_i = \mathbf{prox}_{f_i}(v_i), \quad i = 1, \ldots, N$$

- key to parallel/distributed proximal algorithms
- **example**: if $f = \|\cdot\|_1$, then

$$\mathbf{prox}_{\lambda f}(v) = (v - \lambda)_+ - (-v - \lambda)_+ = \begin{cases} v_i - \lambda & v_i \geq \lambda \\ 0 & |v_i| \leq \lambda \\ v_i + \lambda & v_i \leq -\lambda \end{cases}$$

  a simple elementwise operation called *soft thresholding*

Fixed points

- the point $x^\star$ minimizes f if and only if $x^\star$ is a fixed point:

$$x^\star = \mathbf{prox}_f(x^\star)$$

- provides a link between proximal operators and fixed point theory
- many proximal algorithms can be viewed as methods for finding fixed points of appropriate operators

Moreau-Yosida regularization

- **infimal convolution** of f and g, denoted $f \,\square\, g$, is defined as

$$(f \,\square\, g)(v) = \inf_x \left(f(x) + g(v - x)\right)$$

- **Moreau envelope** or **Moreau-Yosida regularization** of f is

$$M_{\lambda f}(v) = \inf_x \left(f(x) + (1/2\lambda)\|x - v\|_2^2\right)$$

- a smoothed or regularized form of f:
  - always has full domain
  - always continuously differentiable C1
  - has the same minimizers as f

- can minimize M_f instead of f, though M_f could be hard to evaluate

Moreau-Yosida regularization

- motivation: can show that

$$M_f = (f^* + (1/2)\|\cdot\|_2^2)^*$$

- in general, φ^* is smooth when φ is strongly convex
- Moreau envelope obtains a smooth approximation via:
  1. taking conjugate
  2. regularizing to get a strongly convex function
  3. taking conjugate again
- **example**: Moreau envelope of $|\cdot|$ is the Huber function

$$\varphi^{\text{huber}}(x) = \begin{cases} x^2 & |x| \le 1 \\ 2|x| - 1 & |x| > 1 \end{cases}$$

Modified gradient step

- many relationships between proximal operators and gradient steps
- proximal operator is gradient step for Moreau envelope:

$$\mathbf{prox}_{\lambda f}(x) = x - \lambda \nabla M_{\lambda f}(x)$$

- for small λ, $\mathbf{prox}_{\lambda f}$ converges to gradient step in f:

$$\mathbf{prox}_{\lambda f}(x) = x - \lambda \nabla f(x) + o(\lambda)$$

- parameter can be interpreted as a step size, though proximal methods will generally work even for large step sizes, unlike gradient method

Resolvent of subdifferential operator

- if $z = \mathbf{prox}_{\lambda f}(x)$, then

$$z = \underset{u}{\operatorname{argmin}} \left(f(u) + (1/2\lambda)\|u - x\|_2^2 \right)$$

- can rewrite as

$$\begin{aligned} 0 &\in \partial_z \left(f(z) + (1/2\lambda)\|z - x\|_2^2 \right) \\ 0 &\in \partial f(z) + (1/\lambda)(z - x) \\ x &\in (I + \lambda \partial f)(z) = z + \lambda \partial f(z) \\ z &\in (I + \lambda \partial f)^{-1}(x) \end{aligned}$$

- must be careful to interpret ∂f and expressions using it as relations
- mapping $(I + \lambda \partial f)^{-1}$ known as **resolvent** of operator ∂f

Moreau decomposition

- following relation always holds:

$$v = \mathbf{prox}_f(v) + \mathbf{prox}_{f^*}(v)$$

- main link between proximal operators and duality
- a generalization of orthogonal decomposition induced by subspace L:

$$v = \Pi_L(v) + \Pi_{L^\perp}(v)$$

follows from Moreau decomposition and $(I_L)^* = I_{L^\perp}$

Norms and norm balls

- **in general**: if $f = \|\cdot\|$ and $\mathcal{B}$ is unit ball of dual norm, then

$$\mathbf{prox}_{\lambda f}(v) = v - \lambda \Pi_{\mathcal{B}}(v/\lambda)$$

  by Moreau decomposition

- if $f = \|\cdot\|_2$ and $\mathcal{B}$ is the unit ℓ_2 ball, then

$$\Pi_{\mathcal{B}}(v) = \begin{cases} v/\|v\|_2 & \|v\|_2 > 1 \\ v & \|v\|_2 \leq 1 \end{cases}$$

$$\mathbf{prox}_{\lambda f}(v) = \begin{cases} (1 - \lambda/\|v\|_2)v & \|v\|_2 \geq \lambda \\ 0 & \|v\|_2 < \lambda \end{cases}$$

  sometimes called 'block soft thresholding' operator

Norms and norm balls

- if $f = \|\cdot\|_1$ and $\mathcal{B}$ is the unit ℓ_∞ ball, then

$$(\Pi_{\mathcal{B}}(v))_i = \begin{cases} 1 & v_i > 1 \\ v_i & |v_i| \leq 1 \\ -1 & v_i < -1 \end{cases}$$

  lets us derive (elementwise) **soft thresholding**

$$\mathbf{prox}_{\lambda f}(v) = (v - \lambda)_+ - (-v - \lambda)_+ = \begin{cases} v_i - \lambda & v_i \geq \lambda \\ 0 & |v_i| \leq \lambda \\ v_i + \lambda & v_i \leq -\lambda \end{cases}$$

- if $f = \|\cdot\|_\infty$ and $\mathcal{B}$ is unit ℓ_1 ball, simple algorithms available

Matrix functions

- suppose convex $F : \mathbf{R}^{m\times n} \to \mathbf{R}$ is orthogonally invariant:

$$F(QX\tilde{Q}) = F(X)$$

  for all orthogonal Q, $\tilde{Q}$

- then $F = f \circ \sigma$ and

$$\mathbf{prox}_{\lambda F}(A) = U\,\mathbf{diag}(\mathbf{prox}_{\lambda f}(d))V^T$$

  where $A = U\,\mathbf{diag}(d)V^T$ is the SVD of A and $\sigma(A) = d$

- *e.g.*, $F = \|\cdot\|_*$ has $f = \|\cdot\|_1$ so $\mathbf{prox}_{\lambda F}$ is 'singular value thresholding'

Outline

Proximal minimization

- the *proximal minimization* or *proximal point algorithm* is

$$x^{k+1} := \mathbf{prox}_{\lambda f}(x^k)$$

- the simplest proximal method, can be interpreted as
  - gradient method applied to M_f rather than f
  - simple iteration for finding a fixed point of $\mathbf{prox}_f$
- if $f(x) = (1/2)x^T Ax - b^T x$, reduces to iterative refinement for $Ax = b$
- idea originates with Martinet, Moreau, Rockafellar (1970s)
- works for any fixed $\lambda > 0$, or for non-summable λ^k

Operator splitting

- the most useful proximal methods use the idea of **operator splitting**
- these algorithms minimize $f + g$ only using $\mathbf{prox}_f$ or $\mathbf{prox}_g$
- useful when f and g each have useful structure separately
- very simple historical example: alternating projections to find $x \in \mathcal{C} \cap \mathcal{D}$
- literature mostly from 1950s and 1970s

Proximal gradient method

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

f is smooth

$g : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is closed proper convex

- method:
$$x^{k+1} := \mathbf{prox}_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$$
- converges with rate $O(1/k)$ when ∇f is Lipschitz continuous with constant L and step sizes are $\lambda^k = \lambda \in (0, 1/L]$
- special case: projected gradient method (take $g = I_C$)
- traces back to Bruck, Lions, Mercier (1970s)

Proximal gradient method

if L is not known (usually the case), can use the following line search:

---

given x^k, λ^{k-1}, and parameter $\beta \in (0,1)$.

Let $\lambda := \lambda^{k-1}$.

repeat

1. Let $z := \mathbf{prox}_{\lambda g}(x^k - \lambda \nabla f(x^k))$.
2. **break if** $f(z) \leq \hat{f}_\lambda(z, x^k)$.
3. Update $\lambda := \beta\lambda$.

return $\lambda^k := \lambda$, $x^{k+1} := z$.

---

typical value of β is $1/2$, and

$$\hat{f}_\lambda(x,y) = f(y) + \nabla f(y)^T(x-y) + (1/2\lambda)\|x-y\|_2^2$$

Interpretations

- x^+ is solution to

$$\text{minimize} \quad (1/2)\|x^+ - (x - \lambda \nabla f(x))\|_2^2 + \lambda g(x^+)$$

  trade off between minimizing g and being close to gradient step for f

- majorization-minimization method for $f + g$:
  - keep minimizing convex upper bound to objective tight at previous iterate
  - here, use $\hat{f}_\lambda(x, x^k) + g(x)$ as upper bound (when $\lambda \in (0, 1/L]$)

- $0 \in \nabla f(x^\star) + \partial g(x^\star)$ if and only if

$$x^\star = (I + \lambda \partial g)^{-1}(I - \lambda \nabla f)(x^\star)$$

  i.e., $x^\star$ is a fixed point of *forward-backward operator*

Accelerated proximal gradient method

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

f is smooth

$g : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is closed proper convex

- method:

$$\begin{aligned} y^{k+1} &:= x^k + \omega^k \left(x^k - x^{k-1}\right) \\ x^{k+1} &:= \mathbf{prox}_{\lambda^k g}\left(y^{k+1} - \lambda^k \nabla f(y^{k+1})\right) \end{aligned}$$

  works for $\omega^k = k/(k+3)$ and similar line search as before

- faster $O(1/k^2)$ convergence rate, originated with Nesterov (1983)

ADMM

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

$f,\ g : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ are closed proper convex

- method:

$$\begin{aligned} x^{k+1} &:= \mathbf{prox}_{\lambda f}(z^k - u^k) \\ z^{k+1} &:= \mathbf{prox}_{\lambda g}(x^{k+1} + u^k) \\ u^{k+1} &:= u^k + x^{k+1} - z^{k+1} \end{aligned}$$

- basically, always works and has $O(1/k)$ rate in general
- if f and g are both indicators, get a variation on alternating projections
- originates from Gabay, Mercier, Glowinski, Marrocco in 1970s

Outline

Lasso

$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \gamma\|x\|_1$$

with $A \in \mathbf{R}^{m\times n}$ and $\gamma > 0$

- proximal gradient method (similar for accelerated):

$$x^{k+1} := \mathbf{prox}_{\lambda^k\gamma\|\cdot\|_1}(x^k - \lambda^k A^T(Ax^k - b))$$

- faster implementations:
  - parallel matrix-vector multiplication
  - if $n \ll m$, precompute A^TA and A^Tb, then solve smaller lasso problem with $\tilde{A} = (A^TA)^{1/2}$, $\tilde{b} = A^Tb$; effort is then mostly computing $\tilde{A}$, $\tilde{b}$
  - compute A^TA and A^Tb in parallel with one a_i in memory at a time
  - compute entire regularization path with warm starting

- can easily generalize to group lasso, elastic net, GLMs, ...

Lasso

- ADMM:

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

- all the effort is in x-update (solve linear system)

- faster implementations:
  - if A is fat, use matrix inversion lemma in x-update
  - if using direct method, use factorization caching
  - if using iterative method, warm start
  - if $n \ll m$, precompute $A^T A$ and $A^T b$ (possibly in parallel)

Lasso

Method	Iterations	Time (s)	$p^\star$	Error (abs)	Error (rel)
CVX	15	26.53	16.5822	—	—
Proximal gradient	127	0.72	16.5835	0.09	0.01
Accelerated	23	0.15	16.6006	0.26	0.04
ADMM	20	0.07	16.6011	0.18	0.03

Distributed lasso example

- example with **dense** $A \in \mathbf{R}^{400000 \times 8000}$ (roughly 30 GB of data)
  - distributed solver written in C using MPI and GSL
  - no optimization or tuned libraries (like ATLAS, MKL)
  - split into 80 subsystems across 10 (8-core) machines on Amazon EC2

- computation times

loading data	30s
factorization	5m
subsequent ADMM iterations	0.5–2s
lasso solve (about 15 ADMM iterations)	5–6m

Matrix decomposition

- decompose matrix A into sum of 'simple' components:

$$\begin{array}{ll} \text{minimize} & \varphi_1(X_1) + \gamma_2\varphi_2(X_2) + \cdots + \gamma_N\varphi_N(X_N) \\ \text{subject to} & A = X_1 + X_2 + \cdots + X_N \end{array}$$

- penalty functions can include
  - squared Frobenius norm (make X_i small)
  - entrywise ℓ_1 norm (make X_i sparse)
  - sum-{row,column} norm (row/column sparsity)
  - indicator of elementwise constraints (*e.g.*, known values, bounds)
  - indicator of semidefinite cone
  - nuclear norm (make X_i low rank)

- common example: decompose $A = \text{sparse} + \text{low rank}$

Matrix decomposition via ADMM

- splitting:

$$f(X) = \sum_{i=1}^{N} \varphi_i(X_i), \quad g(X) = I_{\mathcal{C}}(X)$$

with

$$X = (X_1, \ldots, X_N), \quad \mathcal{C} = \{(X_1, \ldots, X_N) \mid X_1 + \cdots + X_N = A\}$$

- ADMM simplifies to:

$$\begin{aligned} X_i^{k+1} &:= \mathbf{prox}_{\lambda\varphi_i}(X_i^k - \overline{X}^k + (1/N)A - U^k) \\ U^{k+1} &:= U^k + \overline{X}^{k+1} - (1/N)A \end{aligned}$$

Matrix decomposition results

problem: decompose $A = \text{rank } 4 + \text{sparse} + \text{small Gaussian noise}$

Method	m	n	Iterations	Time (s)
CVX	10	30	15	1.11
ADMM	10	30	45	0.02
CVX	20	50	17	2.54
ADMM	20	50	42	0.03
CVX	40	80	20	108.14
ADMM	40	80	36	0.07
ADMM	100	200	38	0.58
ADMM	500	1000	42	35.56

note: last instance has 1.5M variables and 500K constraints

Multi-period portfolio optimization

$$\text{minimize} \quad \sum_{t=1}^{T} f_t(x_t) + \sum_{t=1}^{T} g_t(x_t - x_{t-1})$$

f_t is risk-adjusted negative return in period t ('stage costs')
g_t is transaction costs

- splitting: put stage costs in one term and transaction costs in the other
- f is separable across time steps, g is separable across assets
- $\mathbf{prox}_f$ involves solving T single-period problems in parallel
- $\mathbf{prox}_g$ involves n problems in parallel that can be solved in $O(T)$ flops

Multi-period portfolio optimization

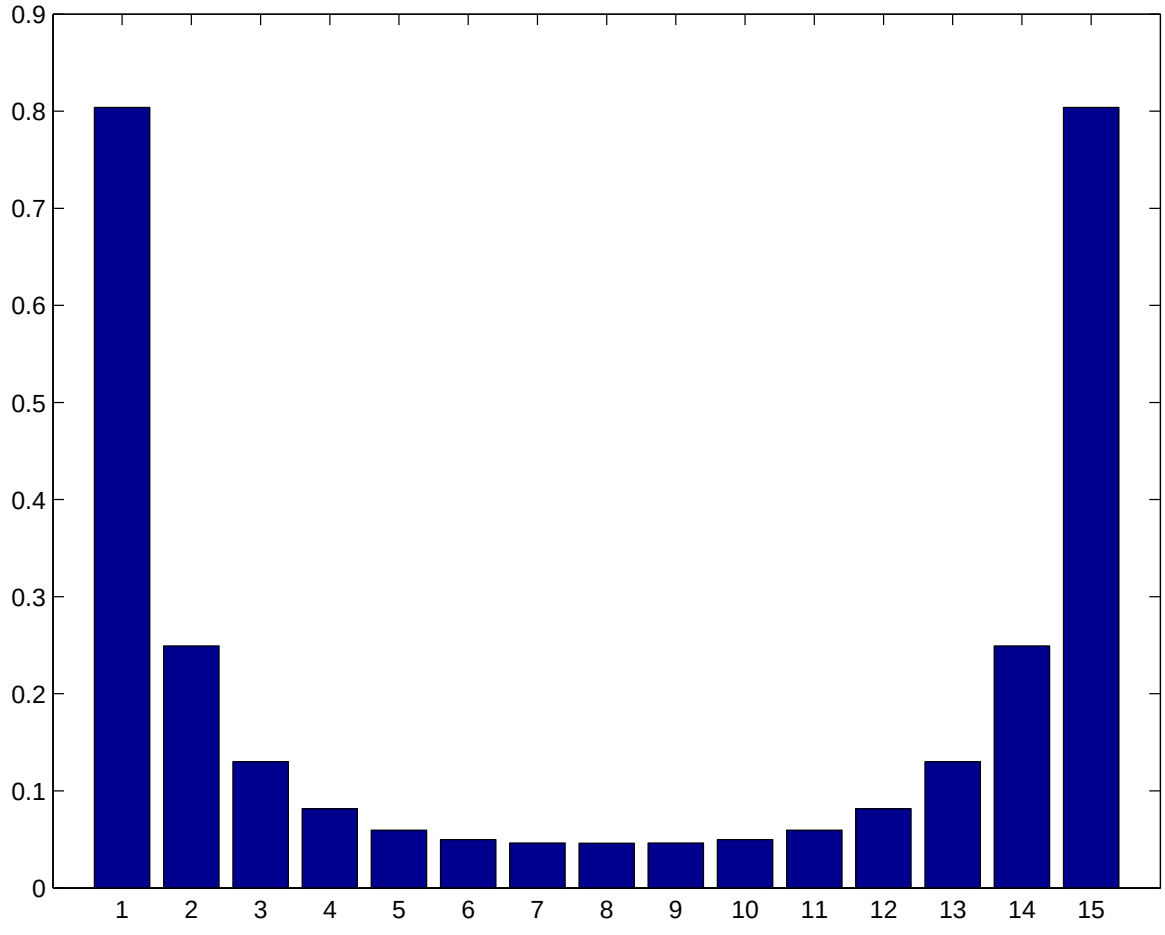

Figure: Time series of ℓ_1 deviation from x^{static}

Multi-period portfolio optimization

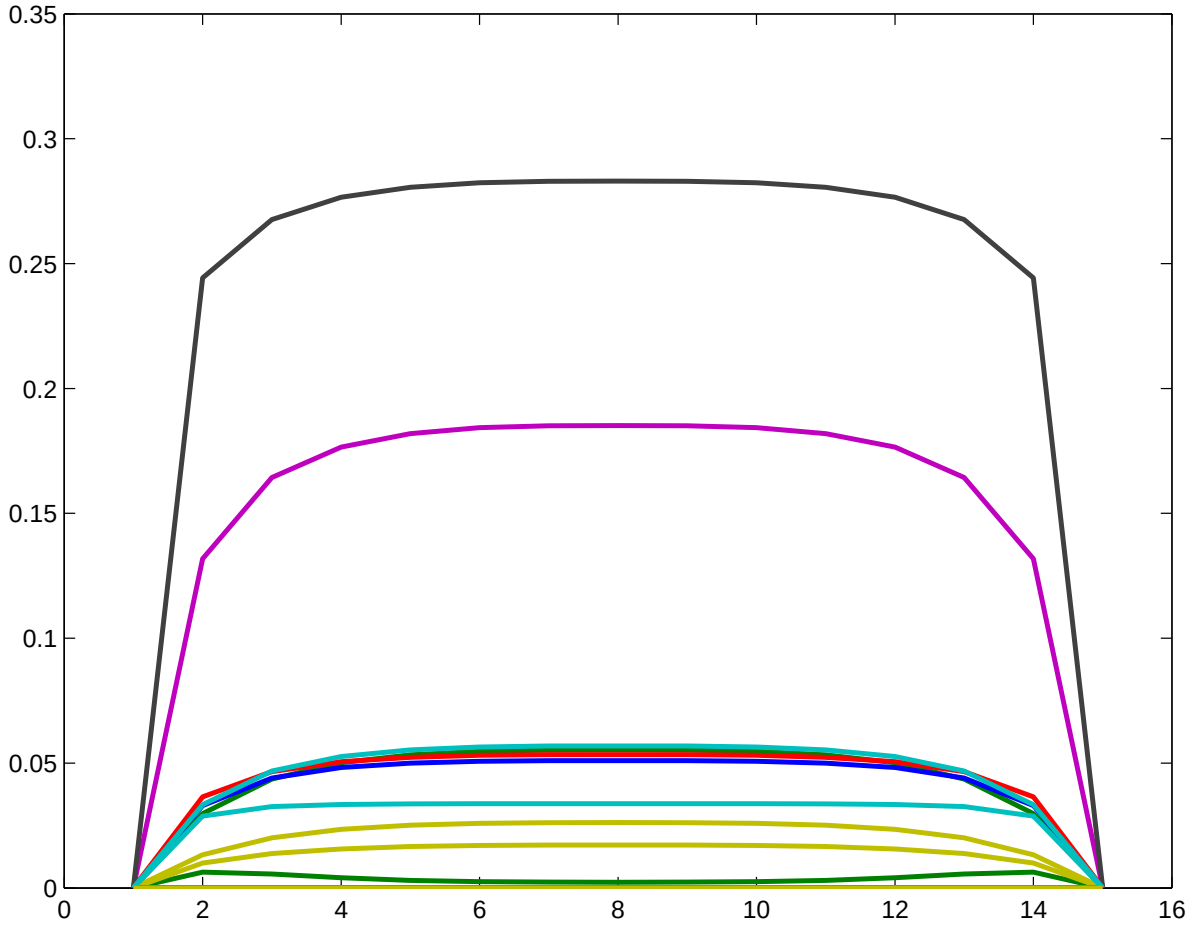

Figure: Time series of asset holdings

Outline

Conclusions

key ingredients:

1. operator splitting
2. proximal operators

often, use problem transformations so when an operator splitting method is used, each part of the problem has simple, small prox operators