

Proximal Gradient Descent (and Acceleration)

Hoàng Nam Dũng

Khoa Toán - Cơ - Tin học, Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội

Last time: subgradient method

Consider the problem

$$\min_x f(x)$$

with f convex, and $\text{dom}(f) = \mathbb{R}^n$.

Subgradient method: choose an initial $x^{(0)} \in \mathbb{R}^n$, and repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot g^{(k-1)}, \quad k = 1, 2, 3, \dots$$

where $g^{(k-1)} \in \partial f(x^{(k-1)})$. We use pre-set rules for the step sizes (e.g., diminishing step sizes rule).

If f is Lipschitz, then subgradient method has a convergence rate $O(1/\varepsilon^2)$.

Upside: very generic. Downside: can be slow — addressed today.

Today

- ▶ Proximal gradient descent
- ▶ Convergence analysis
- ▶ ISTA, matrix completion
- ▶ Special cases
- ▶ Acceleration

Decomposable functions

Suppose

$$f(x) = g(x) + h(x)$$

where

- ▶ g is convex, differentiable, $\text{dom}(g) = \mathbb{R}^n$
- ▶ h is convex, not necessarily differentiable.

If f were differentiable, then gradient descent update would be

$$x^+ = x - t \cdot \nabla f(x)$$

Recall motivation: minimize **quadratic approximation** to f around x , replace $\nabla^2 f(x)$ by $\frac{1}{t}I$

$$x^+ = \operatorname{argmin}_z \underbrace{f(x) + \nabla f(x)^T(z - x) + \frac{1}{2t}\|z - x\|_2^2}_{\tilde{f}_t(z)}.$$

Decomposable functions

In our case f is not differentiable, but $f = g + h$, g differentiable.
Why don't we make quadratic approximation to g , leave h alone?

I.e., update

$$\begin{aligned}x^+ &= \operatorname{argmin}_z \tilde{g}_t(z) + h(z) \\&= \operatorname{argmin}_z g(x) + \nabla g(x)^T (z - x) + \frac{1}{2t} \|z - x\|_2^2 + h(z) \\&= \operatorname{argmin}_z \frac{1}{2t} \|z - (x - t \nabla g(x))\|_2^2 + h(z).\end{aligned}$$

$$\frac{1}{2t} \|z - (x - t \nabla g(x))\|_2^2 + h(z)$$

stay close to gradient update for g
also make h small

Proximal mapping

The proximal mapping (or prox-operator) of a convex function h is defined as

$$\text{prox}_h(x) = \operatorname{argmin}_z \frac{1}{2} \|x - z\|_2^2 + h(z).$$

Proximal mapping

The **proximal mapping** (or **prox-operator**) of a convex function h is defined as

$$\text{prox}_h(x) = \operatorname{argmin}_z \frac{1}{2} \|x - z\|_2^2 + h(z).$$

Examples:

- ▶ $h(x) = 0$: $\text{prox}_h(x) = x$.

Proximal mapping

The **proximal mapping** (or **prox-operator**) of a convex function h is defined as

$$\text{prox}_h(x) = \operatorname{argmin}_z \frac{1}{2} \|x - z\|_2^2 + h(z).$$

Examples:

- ▶ $h(x) = 0$: $\text{prox}_h(x) = x$.
- ▶ $h(x)$ is indicator function of a closed convex set C : prox_h is the projection on C

$$\text{prox}_h(x) = \operatorname{argmin}_{z \in C} \frac{1}{2} \|x - z\|_2^2 = P_C(x).$$

Proximal mapping

The **proximal mapping** (or **prox-operator**) of a convex function h is defined as

$$\text{prox}_h(x) = \operatorname{argmin}_z \frac{1}{2} \|x - z\|_2^2 + h(z).$$

Examples:

- ▶ $h(x) = 0$: $\text{prox}_h(x) = x$.
- ▶ $h(x)$ is indicator function of a closed convex set C : prox_h is the projection on C

$$\text{prox}_h(x) = \operatorname{argmin}_{z \in C} \frac{1}{2} \|x - z\|_2^2 = P_C(x).$$

- ▶ $h(x) = \|x\|_1$: prox_h is the 'soft-threshold' (shrinkage) operation

$$\text{prox}_h(x)_i = \begin{cases} x_i - 1 & x_i \geq 1 \\ 0 & |x_i| \leq 1 \\ x_i + 1 & x_i \leq -1. \end{cases}$$

Proximal mapping

Theorem

If h is convex and closed (has closed epigraph) then

$$\text{prox}_h(x) = \operatorname{argmin}_z \frac{1}{2} \|x - z\|_2^2 + h(z).$$

exists and is unique for all x .

Chứng minh.

See <http://www.seas.ucla.edu/~vandenbe/236C/lectures/proxop.pdf>

Uniqueness since the objective function is strictly convex.



Proximal mapping

Theorem

If h is convex and closed (has closed epigraph) then

$$\text{prox}_h(x) = \operatorname{argmin}_z \frac{1}{2} \|x - z\|_2^2 + h(z).$$

exists and is unique for all x .

Chứng minh.

See <http://www.seas.ucla.edu/~vandenbe/236C/lectures/proxop.pdf>

Uniqueness since the objective function is strictly convex. □

Optimality condition

$$\begin{aligned} z = \text{prox}_h(x) &\Leftrightarrow x - z \in \partial h(z) \\ &\Leftrightarrow h(u) \geq h(z) + (x - z)^T (u - z), \forall u. \end{aligned}$$

Properties of proximal mapping

Theorem

Proximal mappings are *firmly nonexpansive* (co-coercive with constant 1)

$$(\operatorname{prox}_h(x) - \operatorname{prox}_h(y))^T (x - y) \geq \|\operatorname{prox}_h(x) - \operatorname{prox}_h(y)\|_2^2.$$

Properties of proximal mapping

Theorem

Proximal mappings are *firmly nonexpansive* (co-coercive with constant 1)

$$(\operatorname{prox}_h(x) - \operatorname{prox}_h(y))^T(x - y) \geq \|\operatorname{prox}_h(x) - \operatorname{prox}_h(y)\|_2^2.$$

Chứng minh.

With $u = \operatorname{prox}_h(x)$ and $v = \operatorname{prox}_h(y)$ we have

$$x - u \in \partial f(u) \quad \text{and} \quad y - v \in \partial f(v).$$

From the monotonicity of subdifferential we get

$$((x - u) - (y - v))^T(u - v) \geq 0.$$

Properties of proximal mapping

Theorem

Proximal mappings are *firmly nonexpansive* (co-coercive with constant 1)

$$(\text{prox}_h(x) - \text{prox}_h(y))^T(x - y) \geq \|\text{prox}_h(x) - \text{prox}_h(y)\|_2^2.$$

Chứng minh.

With $u = \text{prox}_h(x)$ and $v = \text{prox}_h(y)$ we have

$$x - u \in \partial f(u) \quad \text{and} \quad y - v \in \partial f(v).$$

From the monotonicity of subdifferential we get

$$((x - u) - (y - v))^T(u - v) \geq 0.$$

From firm nonexpansiveness and Cauchy-Schwarz inequality we get *nonexpansiveness* (Lipschitz continuity with constant 1)

$$\|\text{prox}_h(x) - \text{prox}_h(y)\|_2 \leq \|x - y\|_2.$$

Proximal gradient descent

Proximal gradient descent: choose initialize $x^{(0)}$, repeat:

$$x^{(k)} = \text{prox}_{t_k h} (x^{(k-1)} - t_k \cdot \nabla g(x^{(k-1)})), \quad k = 1, 2, 3, \dots$$

Proximal gradient descent

Proximal gradient descent: choose initialize $x^{(0)}$, repeat:

$$x^{(k)} = \text{prox}_{t_k h} (x^{(k-1)} - t_k \cdot \nabla g(x^{(k-1)})), \quad k = 1, 2, 3, \dots$$

To make this update step look familiar, can rewrite it as

$$x^{(k)} = x^{(k-1)} - t_k \cdot G_{t_k}(x^{(k-1)})$$

where G_t is the generalized gradient of f ,

$$G_t(x) = \frac{x - \text{prox}_{th}(x - t\nabla g(x))}{t}.$$

Proximal gradient descent

Proximal gradient descent: choose initialize $x^{(0)}$, repeat:

$$x^{(k)} = \text{prox}_{t_k h} (x^{(k-1)} - t_k \cdot \nabla g(x^{(k-1)})), \quad k = 1, 2, 3, \dots$$

To make this update step look familiar, can rewrite it as

$$x^{(k)} = x^{(k-1)} - t_k \cdot G_{t_k}(x^{(k-1)})$$

where G_t is the generalized gradient of f ,

$$G_t(x) = \frac{x - \text{prox}_{th}(x - t\nabla g(x))}{t}.$$

For $h = 0$ it is gradient descent.

Examples

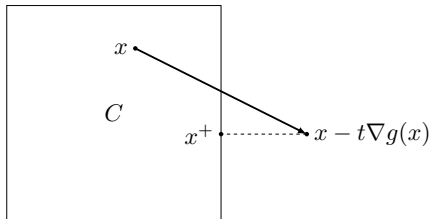
$$\text{minimize } g(x) + h(x)$$

Gradient method: special case with $h(x) = 0$

$$x^+ = x - t\nabla g(x)$$

Gradient projection method: special case with $h(x) = \delta_C(x)$ (indicator of C)

$$x^+ = P_C(x - t\nabla g(x))$$



What good did this do?

You have a right to be suspicious ... may look like we just swapped one minimization problem for another.

Key point is that $\text{prox}_h(\cdot)$ is can be **computed analytically** for a lot of important functions h^1 .

Note:

- ▶ Mapping $\text{prox}_h(\cdot)$ doesn't depend on g at all, only on h .
- ▶ Smooth part g can be complicated, we only need to compute its gradients.

Convergence analysis: will be in terms of number of iterations of the algorithm. Each iteration evaluates $\text{prox}_h(\cdot)$ once and this can be cheap or expensive depending on h .

¹see <http://www.seas.ucla.edu/~vandenbe/236C/lectures/proxop.pdf>

Example: ISTA (Iterative Shrinkage-Thresholding Algorithm)

Given $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, recall lasso criterion

$$f(\beta) = \underbrace{\frac{1}{2} \|y - X\beta\|_2^2}_{g(\beta)} + \underbrace{\lambda \|\beta\|_1}_{h(\beta)}.$$

Proximal mapping is now

$$\begin{aligned} \text{prox}_{th}(\beta) &= \operatorname{argmin}_z \frac{1}{2t} \|\beta - z\|_2^2 + \lambda \|z\|_1 \\ &= S_{\lambda t}(\beta), \end{aligned}$$

where $S_\lambda(\beta)$ is the soft-thresholding operator

$$[S_\lambda(\beta)]_i = \begin{cases} \beta_i - \lambda & \text{if } \beta_i > \lambda \\ 0 & \text{if } -\lambda \leq \beta_i \leq \lambda, \\ \beta_i + \lambda & \text{if } \beta_i < -\lambda \end{cases} \quad i = 1, \dots, n.$$

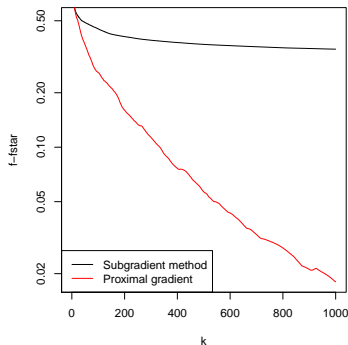
Example: ISTA (Iterative Shrinkage-Thresholding Algorithm)

Recall $\nabla g(\beta) = -X^T(y - X\beta)$, hence proximal gradient update is

$$\beta^+ = S_{\lambda t}(\beta + tX^T(y - X\beta)).$$

Often called the **iterative soft-thresholding algorithm (ISTA)**². Very simple algorithm.

Example of proximal gradient (ISTA) vs. subgradient method convergence rates



²Beck and Teboulle (2008), "A fast iterative shrinkage-thresholding algorithm for linear inverse problems"

Backtracking line search

Backtracking for prox gradient descent works similar as before (in gradient descent), but operates on g and not f .

Choose parameter $0 < \beta < 1$. At each iteration, start at $t = t_{\text{init}}$, and while

$$g(x - tG_t(x)) > g(x) - t\nabla g(x)^T G_t(x) + \frac{t}{2}\|G_t(x)\|_2^2$$

shrink $t = \beta t$, for some $0 < \beta < 1$. Else perform proximal gradient update.

(Alternative formulations exist that require less computation, i.e., fewer calls to prox)

Convergence analysis

For criterion $f(x) = g(x) + h(x)$, we assume

- ▶ g is convex, differentiable, $\text{dom}(g) = \mathbb{R}^n$, and ∇g is Lipschitz continuous with constant $L > 0$.
- ▶ h is convex, $\text{prox}_{th}(x) = \text{argmin}_z \{\|x - z\|_2^2/(2t) + h(z)\}$ can be evaluated.

Theorem

Proximal gradient descent with fixed step size $t \leq 1/L$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}$$

and same result holds for backtracking with t replaced by β/L .

Convergence analysis

For criterion $f(x) = g(x) + h(x)$, we assume

- ▶ g is convex, differentiable, $\text{dom}(g) = \mathbb{R}^n$, and ∇g is Lipschitz continuous with constant $L > 0$.
- ▶ h is convex, $\text{prox}_{th}(x) = \text{argmin}_z \{ \|x - z\|_2^2 / (2t) + h(z) \}$ can be evaluated.

Theorem

Proximal gradient descent with fixed step size $t \leq 1/L$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}$$

and same result holds for backtracking with t replaced by β/L .

Proximal gradient descent has convergence rate $O(1/k)$ or $O(1/\varepsilon)$.
Same as gradient descent! (But remember, prox cost matters ...).

Convergence analysis

For criterion $f(x) = g(x) + h(x)$, we assume

- ▶ g is convex, differentiable, $\text{dom}(g) = \mathbb{R}^n$, and ∇g is Lipschitz continuous with constant $L > 0$.
- ▶ h is convex, $\text{prox}_{th}(x) = \text{argmin}_z \{ \|x - z\|_2^2 / (2t) + h(z) \}$ can be evaluated.

Theorem

Proximal gradient descent with fixed step size $t \leq 1/L$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}$$

and same result holds for backtracking with t replaced by β/L .

Proximal gradient descent has convergence rate $O(1/k)$ or $O(1/\varepsilon)$. Same as gradient descent! (But remember, prox cost matters ...).

Proof: See <http://www.seas.ucla.edu/~vandenbe/236C/lectures/proxgrad.pdf>

Example: matrix completion

Given a matrix $Y \in \mathbb{R}^{m \times n}$, and only observe entries $Y_{ij}, (i, j) \in \Omega$. Suppose we want to fill in missing entries (e.g., for a recommender system), so we solve a **matrix completion** problem³

$$\min_B \frac{1}{2} \sum_{(i,j) \in \Omega} (Y_{ij} - B_{ij})^2 + \lambda \|B\|_{\text{tr}}.$$

Here $\|B\|_{\text{tr}}$ is the trace (or nuclear) norm of B

$$\|B\|_{\text{tr}} = \sum_{i=1}^r \sigma_i(B),$$

where $r = \text{rank}(B)$ and $\sigma_1(X) \geq \dots \geq \sigma_r(X) \geq 0$ are the singular values⁴.

³Wikipedia: In the case of the Netflix problem the ratings matrix is expected to be low-rank since user preferences can often be described by a few factors, such as the movie genre and time of release

⁴<https://math.berkeley.edu/~hutching/teach/54-2017/svd-notes.pdf>

Example: matrix completion

Define P_Ω , projection operator onto observed set

$$[P_\Omega(B)]_{ij} = \begin{cases} B_{ij} & (i,j) \in \Omega \\ 0 & (i,j) \notin \Omega. \end{cases}$$

Then the criterion is

$$f(B) = \underbrace{\frac{1}{2} \|P_\Omega(Y) - P_\Omega(B)\|_F^2}_{g(B)} + \underbrace{\lambda \|B\|_{\text{tr}}}_{h(B)}.$$

Two ingredients needed for proximal gradient descent:

- Gradient calculation

$$\nabla g(B) = -(P_\Omega(Y) - P_\Omega(B)).$$

- Prox function

$$\text{prox}_{th}(B) = \underset{Z}{\operatorname{argmin}} \frac{1}{2t} \|B - Z\|_F^2 + \lambda \|Z\|_{\text{tr}}.$$

Example: matrix completion

Claim:

$$\text{prox}_t(B) = S_{\lambda t}(B), \text{ matrix soft-thresholding at the level } \lambda.$$

Here $S_{\lambda}(B)$ is defined by

$$S_{\lambda}(B) = U\Sigma_{\lambda}V^T$$

where $B = U\Sigma V^T$ is an SVD, and Σ_{λ} is diagonal with

$$(\Sigma_{\lambda})_{ii} = \max\{\Sigma_{ii} - \lambda, 0\}.$$

Proof: note that $\text{prox}_{th}(B) = Z$, where Z satisfies

$$0 \in Z - B + \lambda t \cdot \partial\|Z\|_{\text{tr}}.$$

Helpful fact: if $Z = U\Sigma V^T$, then

$$\partial\|Z\|_{\text{tr}} = \{UV^T + W : \|W\|_{\text{op}} \leq 1, U^T W = 0, W V = 0\}.$$

Now plug in $Z = S_{\lambda t}(B)$ and check that we can get 0.

Example: matrix completion

Hence proximal gradient update step is

$$B^+ = S_{\lambda t} (B + t(P_{\Omega}(Y) - P_{\Omega}(B))).$$

Note that $\nabla g(B)$ is Lipschitz continuous with $L = 1$, so we can choose fixed step size $t = 1$. Update step is now

$$B^+ = S_{\lambda}(P_{\Omega}(Y) + P_{\Omega}^{\perp}(B))$$

where P_{Ω}^{\perp} projects onto unobserved set, $P_{\Omega}(B) + P_{\Omega}^{\perp}(B) = B$.

This is the **soft-impute** algorithm⁵, simple and effective method for matrix completion.

⁵Mazumder et al. (2011), "Spectral regularization algorithms for learning large incomplete matrices"

Special cases

Proximal gradient descent also called composite gradient descent or **generalized gradient descent**.

Why “generalized”? This refers to the several special cases, when minimizing $f = g + h$

- ▶ $h = 0$ – gradient descent
- ▶ $h = I_C$ – projected gradient descent
- ▶ $g = 0$ – proximal minimization algorithm.

Therefore these algorithms all have $O(1/\varepsilon)$ convergence rate.

Projected gradient descent

Given closed, convex set $C \in \mathbb{R}^n$,

$$\min_{x \in C} g(x) \iff \min_x g(x) + I_C(x)$$

where $I_C(x) = \begin{cases} 0 & x \in C \\ \infty & x \notin C \end{cases}$ is the indicator function of C .

Projected gradient descent

Given closed, convex set $C \in \mathbb{R}^n$,

$$\min_{x \in C} g(x) \iff \min_x g(x) + I_C(x)$$

where $I_C(x) = \begin{cases} 0 & x \in C \\ \infty & x \notin C \end{cases}$ is the indicator function of C .

We have

$$\begin{aligned} \text{prox}_{tI_C}(x) &= \operatorname{argmin}_z \frac{1}{2t} \|x - z\|_2^2 + I_C(z) \\ &= \operatorname{argmin}_{z \in C} \|x - z\|_2^2, \end{aligned}$$

i.e., $\text{prox}_{tI_C}(x) = P_C(x)$, projection operator onto C .

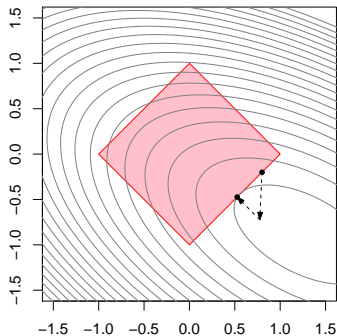
Projected gradient descent

Therefore proximal gradient update step is

$$x^+ = P_C(x - t\nabla g(x)),$$

i.e., perform usual gradient update and then project back onto C .

Called **projected gradient descent**.



Proximal minimization algorithm

Consider for h convex (not necessarily differentiable)

$$\min_x h(x).$$

Proximal gradient update step is just

$$x^+ = \operatorname{argmin}_z \frac{1}{2t} \|x - z\|_2^2 + h(z).$$

Called **proximal minimization algorithm**. Faster than subgradient method, but not implementable unless we know prox in closed form.

What happens if we can't evaluate prox?

Theory for proximal gradient, with $f = g + h$, assumes that prox function can be evaluated, i.e., assumes the minimization

$$\text{prox}_{th}(x) = \operatorname{argmin}_z \frac{1}{2t} \|x - z\|_2^2 + h(z)$$

can be done exactly. In general, not clear what happens if we just minimize this approximately.

But, if you can precisely control the errors in approximating the prox operator, then you can recover the original convergence rates⁶.

In practice, if prox evaluation is done approximately, then it should be done to decently high accuracy.

⁶Schmidt et al. (2011), "Convergence rates of inexact proximal-gradient methods for convex optimization"

Turns out we can accelerate proximal gradient descent in order to achieve the optimal $O(1/\sqrt{\varepsilon})$ convergence rate. Four ideas (three acceleration methods) by Nesterov:

- ▶ 1983: original acceleration idea for smooth functions
- ▶ 1988: another acceleration idea for smooth functions
- ▶ 2005: smoothing techniques for nonsmooth functions, coupled with original acceleration idea
- ▶ 2007: acceleration idea for composite functions⁷.

We will follow Beck and Teboulle (2008), an extension of Nesterov (1983) to composite functions⁸.

⁷Each step uses entire history of previous steps and makes two prox calls

⁸Each step uses information from two last steps and makes one prox call

Accelerated proximal gradient method

As before consider

$$\min_x g(x) + h(x),$$

where g convex, differentiable, and h convex.

Accelerated proximal gradient method: choose initial point $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$, repeat:

$$v = x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)})$$

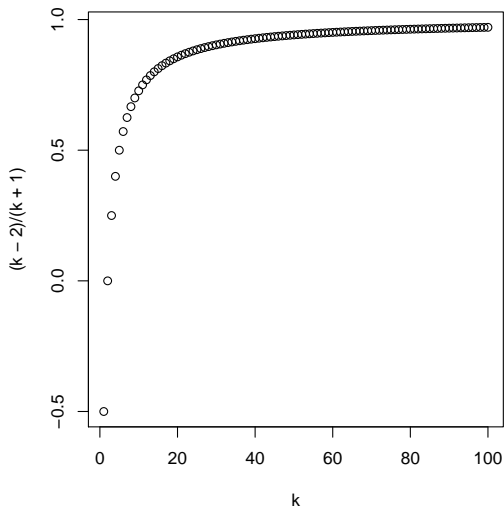
$$x^{(k)} = \text{prox}_{t_k h}(v - t_k \nabla g(v))$$

for $k = 1, 2, 3, \dots$

- ▶ First step $k = 1$ is just usual proximal gradient update.
- ▶ After that, $v = x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)})$ carries some “momentum” from previous iterations.
- ▶ $h = 0$ gives accelerated gradient method.

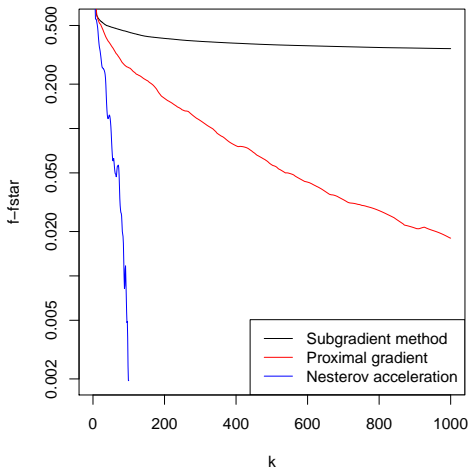
Accelerated proximal gradient method

Momentum weights



Accelerated proximal gradient method

Back to lasso example: acceleration can really help!



Note: accelerated proximal gradient is not a descent method.

Backtracking line search

Backtracking under with acceleration in different ways.

Simple approach: fix $\beta < 1$, $t_0 = 1$. At iteration k , start with $t = t_{k-1}$, and while

$$g(x^+) > g(v) + \nabla g(v)^T(x^+ - v) + \frac{1}{2t}\|x^+ - v\|_2^2$$

shrink $t = \beta t$, and let $x^+ = \text{prox}_{th}(v - t\nabla g(v))$. Else keep x^+ .

Note that this strategy forces us to take decreasing step sizes ... (more complicated strategies exist which avoid this).

Convergence analysis

For criterion $f(x) = g(x) + h(x)$, we assume as before

- ▶ g is convex, differentiable, $\text{dom}(g) = \mathbb{R}^n$, and ∇g is Lipschitz continuous with constant $L > 0$.
- ▶ h is convex, $\text{prox}_{th}(x) = \text{argmin}_z \{ \|x - z\|_2^2 / (2t) + h(z) \}$ can be evaluated.

Theorem

Accelerated proximal gradient method with fixed step size $t \leq 1/L$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{2\|x^{(0)} - x^*\|_2^2}{t(k+1)^2}$$

and same result holds for backtracking, with t replaced by β/L .

Achieves optimal rate $O(1/k^2)$ or $O(1/\sqrt{\varepsilon})$ for first-order methods.

FISTA (Fast ISTA)

Back to lasso problem

$$\min_{\beta} \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1.$$

Recall ISTA (Iterative Soft-thresholding Algorithm):

$$\beta^{(k)} = S_{\lambda t_k}(\beta^{(k-1)} + t_k X^T(y - X\beta^{(k-1)})), \quad k = 1, 2, 3, \dots$$

$S_{\lambda}(\cdot)$ being vector soft-thresholding.

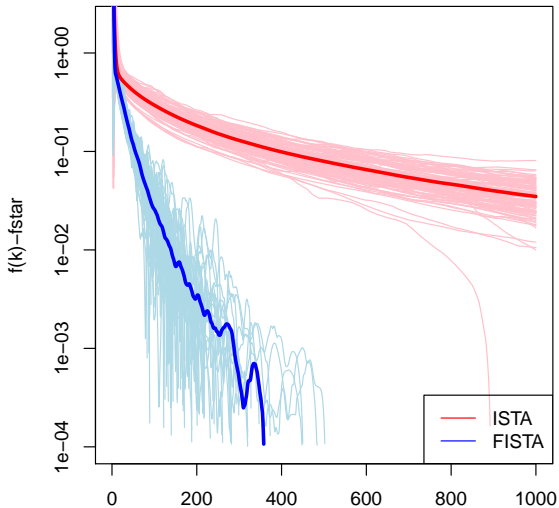
Applying acceleration gives us **FISTA** (F is for Fast)⁹: for $k = 1, 2, 3, \dots$

$$\begin{aligned} v &= \beta^{(k-1)} + \frac{k-2}{k+1}(\beta^{(k-1)} - \beta^{(k-2)}) \\ \beta^{(k)} &= S_{\lambda t_k}(v + t_k X^T(y - Xv)). \end{aligned}$$

⁹Beck and Teboulle (2008) actually call their general acceleration technique (for general g, h) FISTA, which may be somewhat confusing

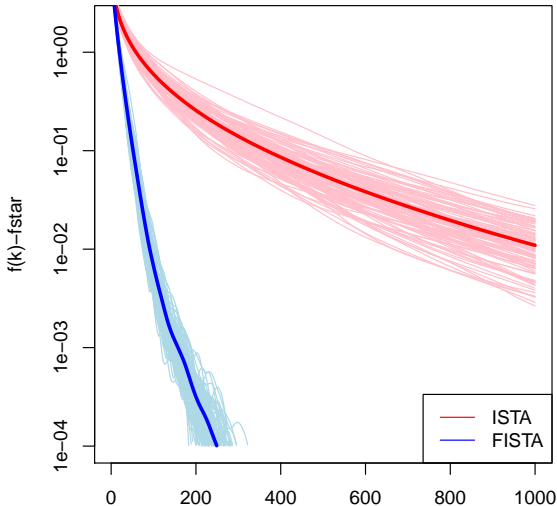
ISTA vs. FISTA

Lasso regression: 100 instances (with $n = 100, p = 500$):



ISTA vs. FISTA

Lasso logistic regression: 100 instances ($n = 100, p = 500$):



Is acceleration always useful?

Acceleration can be a very effective speedup tool ... but should it always be used?

Is acceleration always useful?

Acceleration can be a very effective speedup tool ... but should it always be used?

In practice the speedup of using acceleration is diminished in the presence of **warm starts**. E.g., suppose want to solve lasso problem for tuning parameters values

$$\lambda_1 > \lambda_2 > \cdots > \lambda_r.$$

- ▶ When solving for λ_1 , initialize $x^{(0)} = 0$, record solution $\hat{x}(\lambda_1)$.
- ▶ When solving for λ_j , initialize $x^{(0)} = \hat{x}(\lambda_{j-1})$, the recorded solution for λ_{j-1} .

Over a fine enough grid of λ values, proximal gradient descent can often perform just as well without acceleration.

Is acceleration always useful?

Sometimes backtracking and acceleration can be **disadvantageous**!

Recall matrix completion problem: the proximal gradient update is

$$B^+ = S_\lambda \left(B + t(P_\Omega(Y) - P_\Omega^\perp(B)) \right)$$





where S_λ is the matrix soft-thresholding operator ... requires SVD.

- ▶ One backtracking loop evaluates generalized gradient $G_t(x)$, i.e., evaluates $\text{prox}_t(x)$, across various values of t . For matrix completion, this means multiple SVDs ...
- ▶ Acceleration changes argument we pass to prox: $v - t\nabla g(v)$ instead of $x - t\nabla g(x)$. For matrix completion (and $t = 1$),




$$B - \nabla g(B) = \underbrace{P_\Omega(Y)}_{\text{sparse}} + \underbrace{P_\Omega^\perp(B)}_{\text{low rank}} \Rightarrow \text{fast SVD}$$

$$V - \nabla g(V) = \underbrace{P_\Omega(Y)}_{\text{sparse}} + \underbrace{P_\Omega^\perp(V)}_{\text{not necessarily low rank}} \Rightarrow \text{slow SVD.}$$




Nesterov's four ideas (three acceleration methods):

-  Y. Nesterov (1983), *A method for solving a convex programming problem with convergence rate $O(1/k^2)$*
-  Y. Nesterov (1988), *On an approach to the construction of optimal methods of minimization of smooth convex functions*
-  Y. Nesterov (2005), *Smooth minimization of non-smooth functions*
-  Y. Nesterov (2007), *Gradient methods for minimizing composite objective function*

Extensions and/or analyses:

-  A. Beck and M. Teboulle (2008), *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*
-  S. Becker and J. Bobin and E. Candes (2009), *NESTA: a fast and accurate first-order method for sparse recovery*
-  P. Tseng (2008), *On accelerated proximal gradient methods for convex-concave optimization*

Helpful lecture notes/books:

-  E. Candes, *Lecture notes for Math 301*, Stanford University, Winter 2010-2011
-  Y. Nesterov (1998), *Introductory lectures on convex optimization: a basic course*, Chapter 2
-  L. Vandenberghe, *Lecture notes for EE 236C*, UCLA, Spring 2011-2012