Nonsmooth optimization: Subgradient Method, Proximal Gradient Method

Yu-Xiang Wang
CS292A

(Based on Ryan Tibshirani's 10-725)

Last last time: gradient descent

Consider the problem

$$\min_x \ f(x)$$

for f convex and differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$. Gradient descent: choose initial $x^{(0)} \in \mathbb{R}^n$, repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Step sizes t_k chosen to be fixed and small, or by backtracking line search

If ∇f Lipschitz, gradient descent has convergence rate $O(1/\epsilon)$

Downsides:

- Requires f differentiable — subgradient method
- Can be slow to converge — proximal gradient method

Last time: Subgradient

- Subgradient definition and examples

$$f(y) \geq f(x) + \partial f^T(y - x), \forall x, y$$

- Subdifferential Calculus
  1. Positive scaling, Additive
  2. Affine Composition
  3. Pointwise maximum/supremum
- First order optimality condition:

$$0 \in \partial f(x) \Leftrightarrow x \in \operatorname*{argmin}_x f(x)$$

Outline

Today:

- Subgradient method
- Convergence analysis
- Proximal gradient descent
- Convergence analysis
- ISTA, matrix completion
- Acceleration

Subgradient method

Now consider f convex, having $\mathrm{dom}(f) = \mathbb{R}^n$, but not necessarily differentiable

Subgradient method: like gradient descent, but replacing gradients with subgradients. I.e., initialize $x^{(0)}$, repeat:

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

where $g^{(k-1)} \in \partial f(x^{(k-1)})$, any subgradient of f at $x^{(k-1)}$

Subgradient method is not necessarily a descent method, so we keep track of best iterate $x^{(k)}_{\text{best}}$ among $x^{(0)}, \ldots x^{(k)}$ so far, i.e.,

$$f(x^{(k)}_{\text{best}}) = \min_{i=0,\ldots k} f(x^{(i)})$$

Step size choices

- Fixed step sizes: $t_k = t$ all $k = 1, 2, 3, \ldots$
- Diminishing step sizes: choose to meet conditions

$$\sum_{k=1}^{\infty} t_k^2 < \infty, \quad \sum_{k=1}^{\infty} t_k = \infty,$$

  i.e., square summable but not summable. Important here that step sizes go to zero, but not too fast

There are several other options too, but key difference to gradient descent: step sizes are pre-specified, not adaptively computed

Convergence analysis

Assume that f convex, $\mathrm{dom}(f) = \mathbb{R}^n$, and also that f is Lipschitz continuous with constant $G > 0$, i.e.,

$$|f(x) - f(y)| \leq G\|x - y\|_2 \quad \text{for all } x, y$$

Theorem: For a fixed step size t, subgradient method satisfies

$$\lim_{k \to \infty} f(x_{\text{best}}^{(k)}) \leq f^\star + G^2 t/2$$

Theorem: For diminishing step sizes, subgradient method satisfies

$$\lim_{k \to \infty} f(x_{\text{best}}^{(k)}) = f^\star$$

Basic inequality

Can prove both results from same basic inequality. Key steps:

- Using definition of subgradient,

$$\|x^{(k)} - x^\star\|_2^2 \leq \|x^{(k-1)} - x^\star\|_2^2 - 2t_k\big(f(x^{(k-1)}) - f(x^\star)\big) + t_k^2\|g^{(k-1)}\|_2^2$$

- Iterating last inequality,

$$\|x^{(k)} - x^\star\|_2^2 \leq \|x^{(0)} - x^\star\|_2^2 - 2\sum_{i=1}^k t_i\big(f(x^{(i-1)}) - f(x^\star)\big) + \sum_{i=1}^k t_i^2\|g^{(i-1)}\|_2^2$$

- Using $\|x^{(k)} - x^\star\|_2 \geq 0$, and letting $R = \|x^{(0)} - x^\star\|_2$,

$$0 \leq R^2 - 2\sum_{i=1}^{k} t_i \big(f(x^{(i-1)}) - f(x^\star)\big) + G^2 \sum_{i=1}^{k} t_i^2$$

- Introducing $f(x^{(k)}_{\text{best}}) = \min_{i=0,\ldots k} f(x^{(i)})$, and rearranging, we have the basic inequality

$$f(x^{(k)}_{\text{best}}) - f(x^\star) \leq \frac{R^2 + G^2 \sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k} t_i}$$

For different step sizes choices, convergence results can be directly obtained from this bound. E.g., theorems for fixed and diminishing step sizes follow

Convergence rate

The basic inequality tells us that after k steps, we have

$$f(x_{\text{best}}^{(k)}) - f(x^\star) \leq \frac{R^2 + G^2 \sum_{i=1}^k t_i^2}{2 \sum_{i=1}^k t_i}$$

With fixed step size t, this gives

$$f(x_{\text{best}}^{(k)}) - f^\star \leq \frac{R^2}{2kt} + \frac{G^2 t}{2}$$

For this to be $\leq \epsilon$, let's make each term $\leq \epsilon/2$. So we can choose $t = \epsilon/G^2$, and $k = R^2/t \cdot 1/\epsilon = R^2 G^2/\epsilon^2$

I.e., subgradient method has convergence rate $O(1/\epsilon^2)$ ... compare this to $O(1/\epsilon)$ rate of gradient descent

Example: regularized logistic regression

Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$ for $i = 1, \ldots n$, the logistic regression loss is

$$f(\beta) = \sum_{i=1}^{n} \Big(- y_i x_i^T \beta + \log(1 + \exp(x_i^T \beta)) \Big)$$

This is a smooth and convex, with

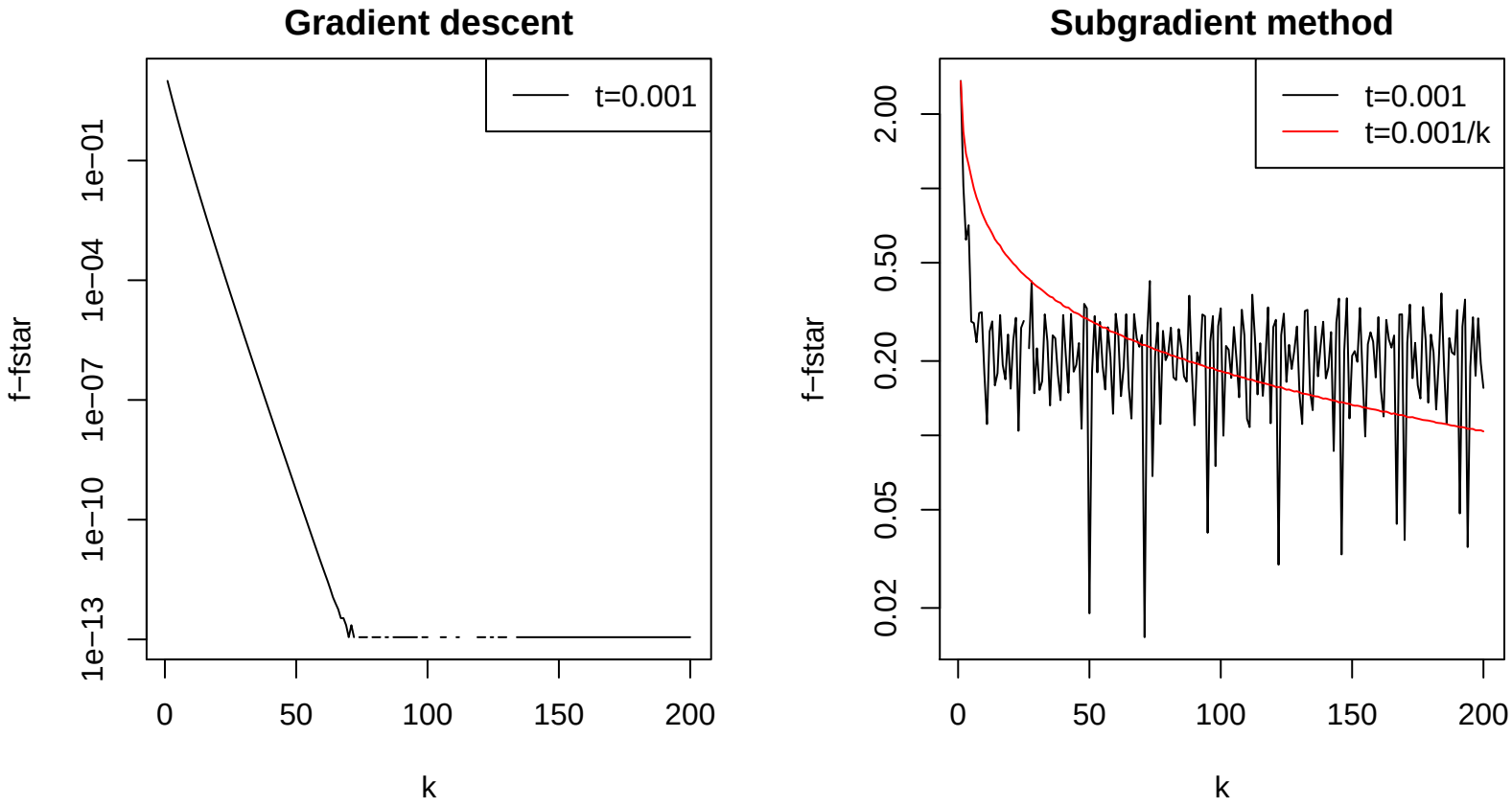
$$\nabla f(\beta) = \sum_{i=1}^{n} \big(y_i - p_i(\beta)\big) x_i$$

where $p_i(\beta) = \exp(x_i^T \beta)/(1 + \exp(x_i^T \beta))$, $i = 1, \ldots n$. Consider the regularized problem:

$$\min_{\beta} \; f(\beta) + \lambda \cdot P(\beta)$$

where $P(\beta) = \|\beta\|_2^2$, ridge penalty; or $P(\beta) = \|\beta\|_1$, lasso penalty

Ridge: use gradients; lasso: use subgradients. Example here has $n = 1000$, $p = 20$:



Step sizes hand-tuned to be favorable for each method (of course comparison is imperfect, but it reveals the convergence behaviors)

Polyak step sizes

Polyak step sizes: when the optimal value $f^\star$ is known, take

$$t_k = \frac{f(x^{(k-1)}) - f^\star}{\|g^{(k-1)}\|_2^2}, \quad k = 1, 2, 3, \ldots$$

Can be motivated from first step in subgradient proof:

$$\|x^{(k)} - x^\star\|_2^2 \leq \|x^{(k-1)} - x^\star\|_2^2 - 2t_k\big(f(x^{(k-1)}) - f(x^\star)\big) + t_k^2\|g^{(k-1)}\|_2^2$$

Polyak step size minimizes the right-hand side

With Polyak step sizes, can show subgradient method converges to optimal value. Convergence rate is still $O(1/\epsilon^2)$

Example: intersection of sets

Suppose we want to find $x^\star \in C_1 \cap \ldots \cap C_m$, i.e., find a point in intersection of closed, convex sets $C_1, \ldots C_m$

First define

$$f_i(x) = \mathrm{dist}(x, C_i), \quad i = 1, \ldots m$$
$$f(x) = \max_{i=1,\ldots m} f_i(x)$$

and now solve

$$\min_x \ f(x)$$

Check: is this convex?

Note that $f^\star = 0 \iff x^\star \in C_1 \cap \ldots \cap C_m$

Recall the distance function $\mathrm{dist}(x, C) = \min_{y \in C} \ \|y - x\|_2$. Last time we computed its gradient

$$\nabla \mathrm{dist}(x, C) = \frac{x - P_C(x)}{\|x - P_C(x)\|_2}$$

where $P_C(x)$ is the projection of x onto C

Also recall subgradient rule: if $f(x) = \max_{i=1,\ldots m} \ f_i(x)$, then

$$\partial f(x) = \mathrm{conv}\left(\bigcup_{i: f_i(x) = f(x)} \partial f_i(x) \right)$$

So if $f_i(x) = f(x)$ and $g_i \in \partial f_i(x)$, then $g_i \in \partial f(x)$

Put these two facts together for intersection of sets problem, with $f_i(x) = \mathrm{dist}(x, C_i)$: if C_i is farthest set from x (so $f_i(x) = f(x)$), and

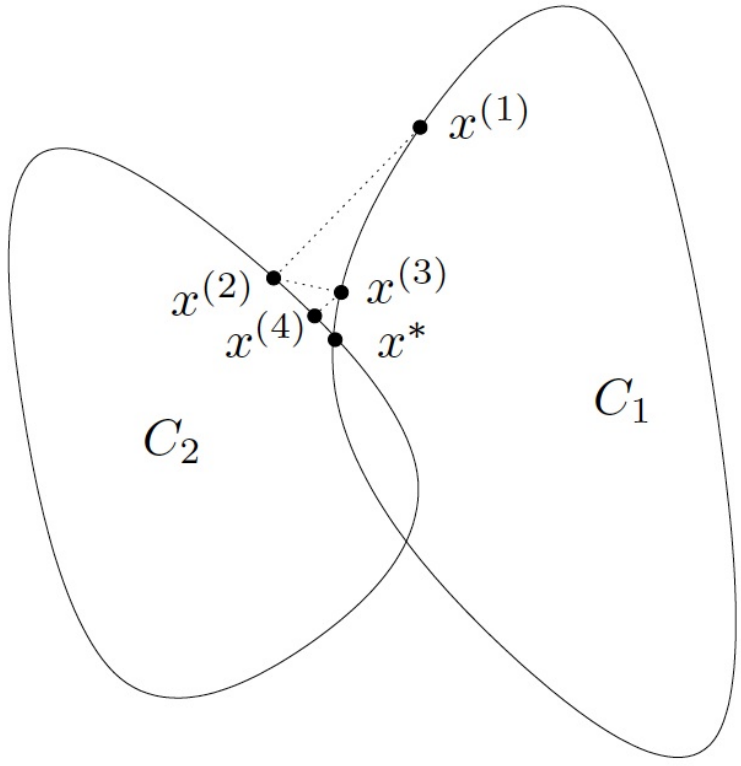
$$g_i = \nabla f_i(x) = \frac{x - P_{C_i}(x)}{\|x - P_{C_i}(x)\|_2}$$

then $g_i \in \partial f(x)$

Now apply subgradient method, with Polyak size $t_k = f(x^{(k-1)})$. At iteration k, with C_i farthest from $x^{(k-1)}$, we perform update

$$\begin{aligned} x^{(k)} &= x^{(k-1)} - f(x^{(k-1)}) \frac{x^{(k-1)} - P_{C_i}(x^{(k-1)})}{\|x^{(k-1)} - P_{C_i}(x^{(k-1)})\|_2} \\ &= P_{C_i}(x^{(k-1)}) \end{aligned}$$

For two sets, this is the famous alternating projections algorithm[1], i.e., just keep projecting back and forth



(From Boyd's lecture notes)

[1] von Neumann (1950), "Functional operators, volume II: The geometry of orthogonal spaces"

Projected subgradient method

To optimize a convex function f over a convex set C,

$$\min_x \ f(x) \ \text{ subject to } \ x \in C$$

we can use the projected subgradient method. Just like the usual subgradient method, except we project onto C at each iteration:

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

Assuming we can do this projection, we get the same convergence guarantees as the usual subgradient method, with the same step size choices

What sets C are easy to project onto? Lots, e.g.,

- Affine images: $\{Ax + b : x \in \mathbb{R}^n\}$
- Solution set of linear system: $\{x : Ax = b\}$
- Nonnegative orthant: $\mathbb{R}^n_+ = \{x : x \geq 0\}$
- Some norm balls: $\{x : \|x\|_p \leq 1\}$ for $p = 1, 2, \infty$
- Some simple polyhedra and simple cones

Warning: it is easy to write down seemingly simple set C, and P_C can turn out to be very hard! E.g., generally hard to project onto arbitrary polyhedron $C = \{x : Ax \leq b\}$

Note: projected gradient descent works too, more next time ...

Can we do better?

Upside of the subgradient method: broad applicability. Downside: $O(1/\epsilon^2)$ convergence rate over problem class of convex, Lipschitz functions is really slow

Nonsmooth first-order methods: iterative methods updating $x^{(k)}$ in

$$x^{(0)} + \mathrm{span}\{g^{(0)}, g^{(1)}, \ldots g^{(k-1)}\}$$

where subgradients $g^{(0)}, g^{(1)}, \ldots g^{(k-1)}$ come from weak oracle

Theorem (Nesterov): For any $k \leq n-1$ and starting point $x^{(0)}$, there is a function in the problem class such that any nonsmooth first-order method satisfies

$$f(x^{(k)}) - f^\star \geq \frac{RG}{2(1+\sqrt{k+1})}$$

What if we assume strong convexity?

Are there functions that are non-smooth but strongly convex?

Theorem: Let f be m-strongly convex and G-Lipschitz. Choose $t_k = \frac{2}{m(k+1)}$, subgradient method satisfies

$$\min_k f(x_k) - f^* \leq \frac{2G^2}{mk}$$

Proof: By the definition of Strong Convexity, we can obtain a

stronger inequality that gets rid of the positive t^2G^2 term when the stepsize t is appropriately chosen.

See (Lacoste-Julien, Schmidt and Bach, 2012) for details.

The bound is also optimal.

Improving on the subgradient method

In words, we cannot do better than the $O(1/\epsilon^2)$ rate of subgradient method for Lipschitz, convex functions.
We cannot do better than $O(1/\epsilon)$ rate for Lipschitz and strongly convex Functions. (unless we go beyond nonsmooth first-order methods).

So instead of trying to improve across the board, we will focus on minimizing composite functions of the form

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

where g is convex and differentiable, h is convex and nonsmooth but "simple"

For a lot of problems (i.e., functions h), we can recover the $O(1/\epsilon)$ rate of gradient descent with a simple algorithm, having important practical consequences

Decomposable functions

Suppose

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

- g is convex, differentiable, $\mathrm{dom}(g) = \mathbb{R}^n$
- 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 $\frac{1}{t} I$,

$$x^+ = \underset{z}{\mathrm{argmin}} \underbrace{f(x) + \nabla f(x)^T (z - x) + \frac{1}{2t} \|z - x\|_2^2}_{\widetilde{f}_t(z)}$$

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

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

$\frac{1}{2t}\big\|z - \big(x - t\nabla g(x)\big)\big\|_2^2$ stay close to gradient update for g

$h(z)$ also make h small

Proximal gradient descent

Define proximal mapping:

$$\text{prox}_t(x) = \underset{z}{\text{argmin}} \ \frac{1}{2t}\|x - z\|_2^2 + h(z)$$

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

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

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, (Nesterov's Gradient Mapping!)

$$G_t(x) = \frac{x - \text{prox}_t\big(x - t\nabla g(x)\big)}{t}$$

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 $\mathrm{prox}_t(\cdot)$ is can be computed analytically for a lot of important functions h. Note:

- Mapping $\mathrm{prox}_t(\cdot)$ 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 $\mathrm{prox}_t(\cdot)$ once, and this can be cheap or expensive, depending on h!

(Similar to Gradient Descent. You will prove it in HW2!)

Example: ISTA

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

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

Prox mapping is now

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

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

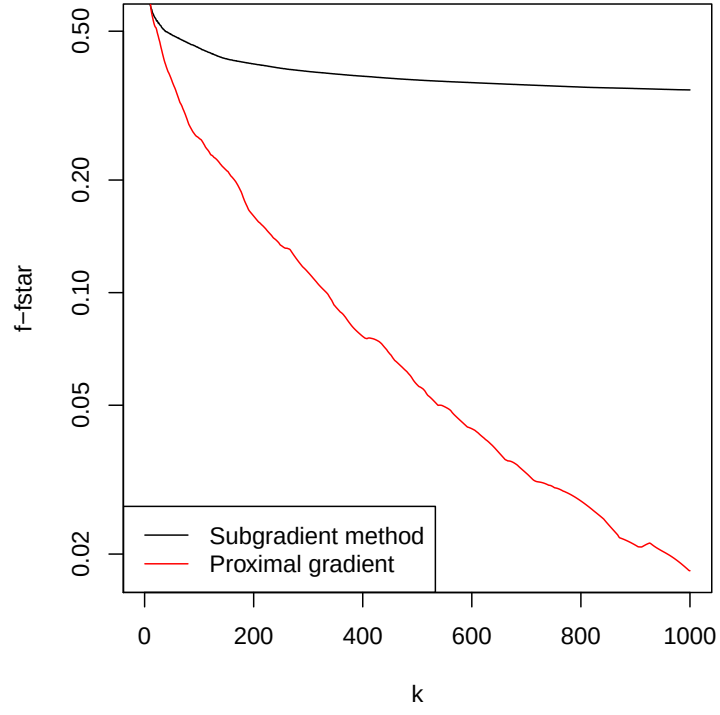
$$[S_\lambda(\beta)]_i = \begin{cases} \beta_i - \lambda & \text{if } \beta_i > \lambda \\ 0 & \text{if } -\lambda \leq \beta_i \leq \lambda \\ \beta_i + \lambda & \text{if } \beta_i < -\lambda \end{cases}, \quad i = 1, \ldots n$$

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

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

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

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



[2] Beck 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

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

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

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

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

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

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

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 problem:

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

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 $\sigma_1(X) \geq \ldots \geq \sigma_r(X) \geq 0$ are the singular values

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:

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

Claim: $\mathrm{prox}_t(B) = S_{\lambda t}(B)$, matrix soft-thresholding at the level λ. 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 $\mathrm{prox}_t(B) = Z$, where Z satisfies

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

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

$$\partial\|Z\|_{\mathrm{tr}} = \{UV^T + W : \|W\|_{\mathrm{op}} \leq 1,\ U^TW = 0,\ WV = 0\}$$

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

Hence proximal gradient update step is:

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

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

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

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

This is the soft-impute algorithm[3], simple and effective method for matrix completion

[3] Mazumder 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 point algorithm

Therefore these algorithms all have $O(1/\epsilon)$ 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

Hence

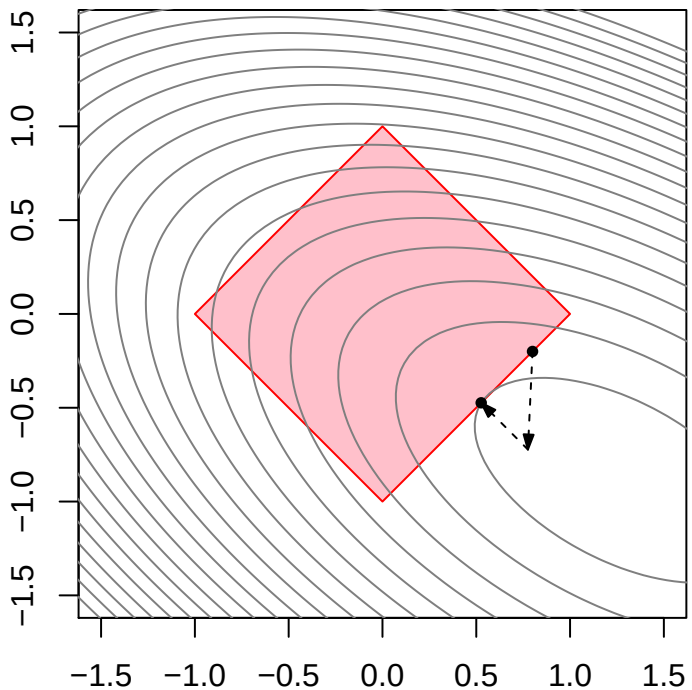
$$\begin{aligned} \text{prox}_t(x) &= \underset{z}{\text{argmin}} \; \frac{1}{2t}\|x - z\|_2^2 + I_C(z) \\ &= \underset{z \in C}{\text{argmin}} \; \|x - z\|_2^2 \end{aligned}$$

I.e., $\text{prox}_t(x) = P_C(x)$, projection operator onto C

Therefore proximal gradient update step is:

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

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



Proximal point 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

$$\text{prox}_t(x) = \underset{z}{\text{argmin}} \ \frac{1}{2t}\|x - z\|_2^2 + h(z)$$

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 rates[4]

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

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

Acceleration

Turns out we can accelerate proximal gradient descent in order to achieve the optimal $O(1/\sqrt{\epsilon})$ 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 functions[5]

We will follow Beck and Teboulle (2008), an extension of Nesterov (1983) to composite functions[6]

[5] Each step uses entire history of previous steps and makes two prox calls

[6] Each step uses information from two last steps and makes one prox call

Accelerated proximal gradient method

As before, consider:

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

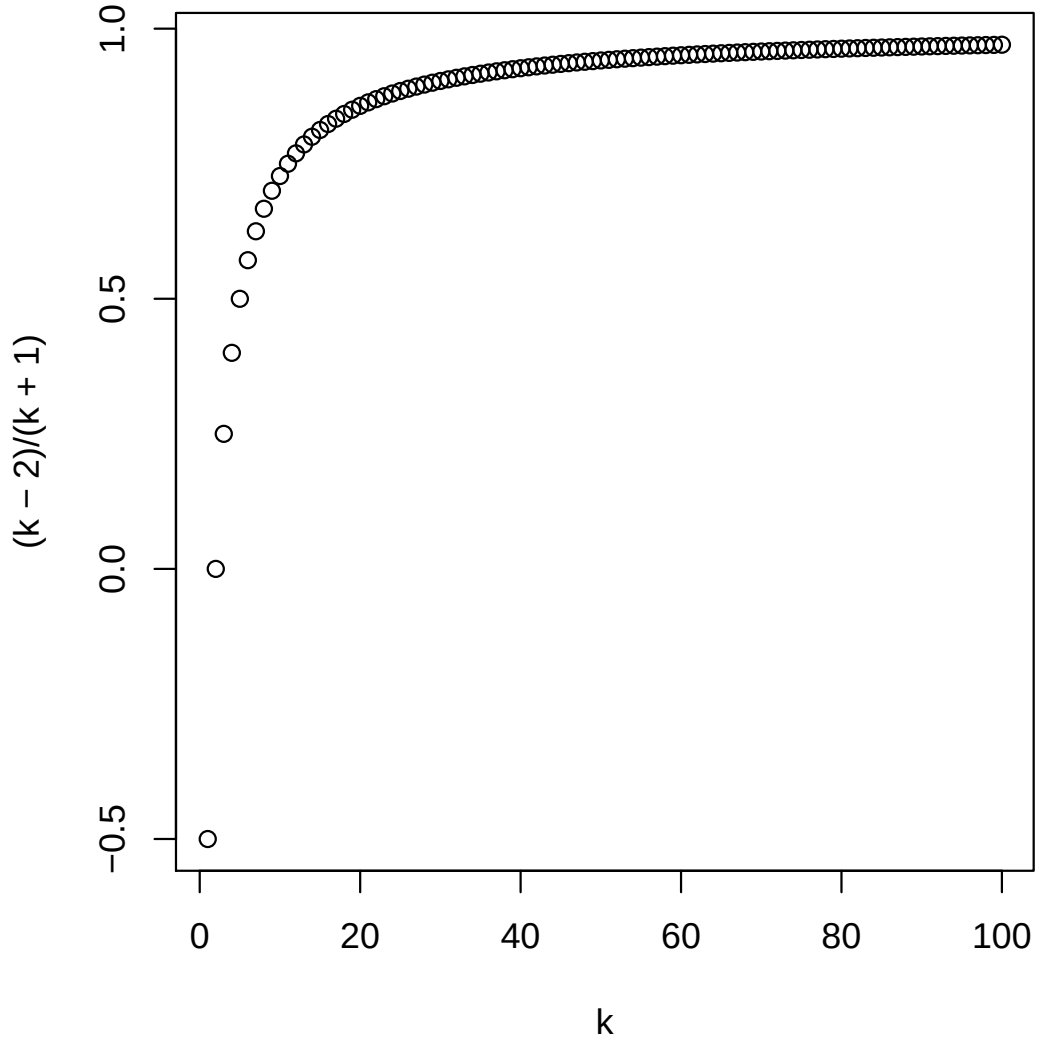
where g convex, differentiable, and h convex. Accelerated proximal gradient method: choose initial point $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$, repeat:

$$\begin{aligned} v &= x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)}) \\ x^{(k)} &= \mathrm{prox}_{t_k}\big(v - t_k \nabla g(v)\big) \end{aligned}$$

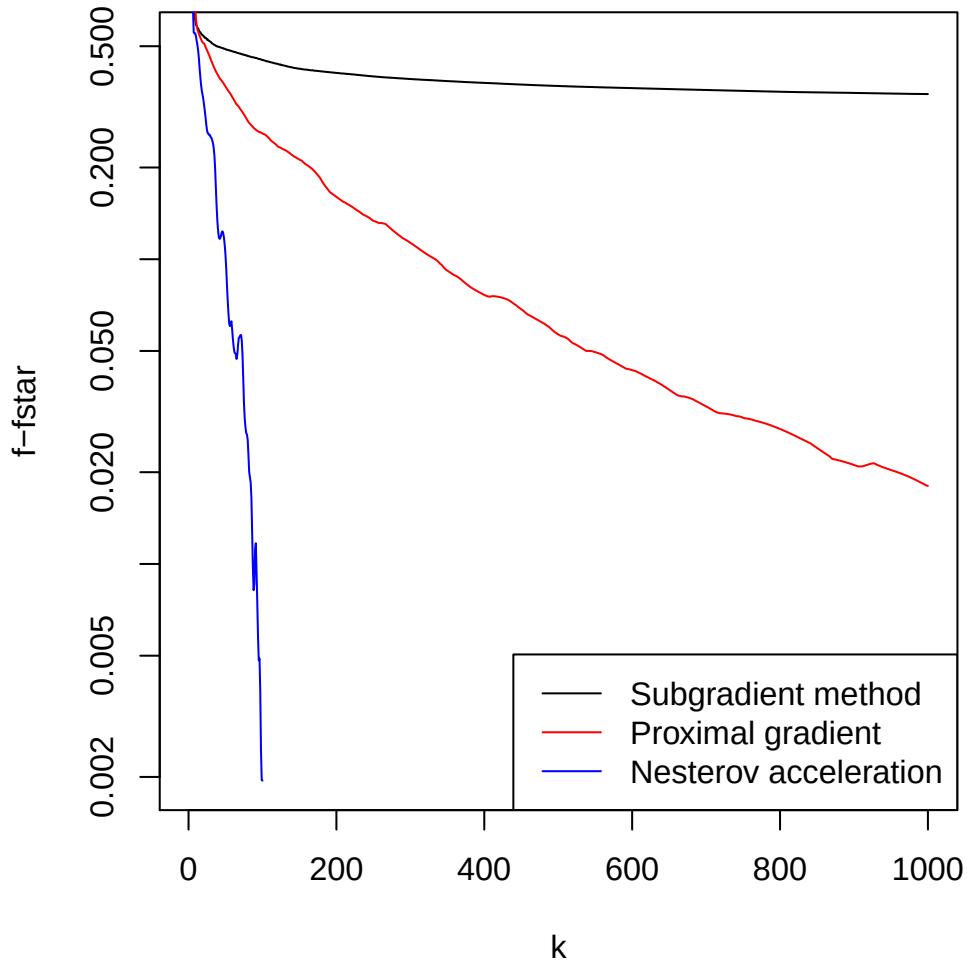
for $k = 1, 2, 3, \ldots$

- 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

Momentum weights:



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

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

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

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

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

FISTA

Back to lasso problem:

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

Recall ISTA (Iterative Soft-thresholding Algorithm):

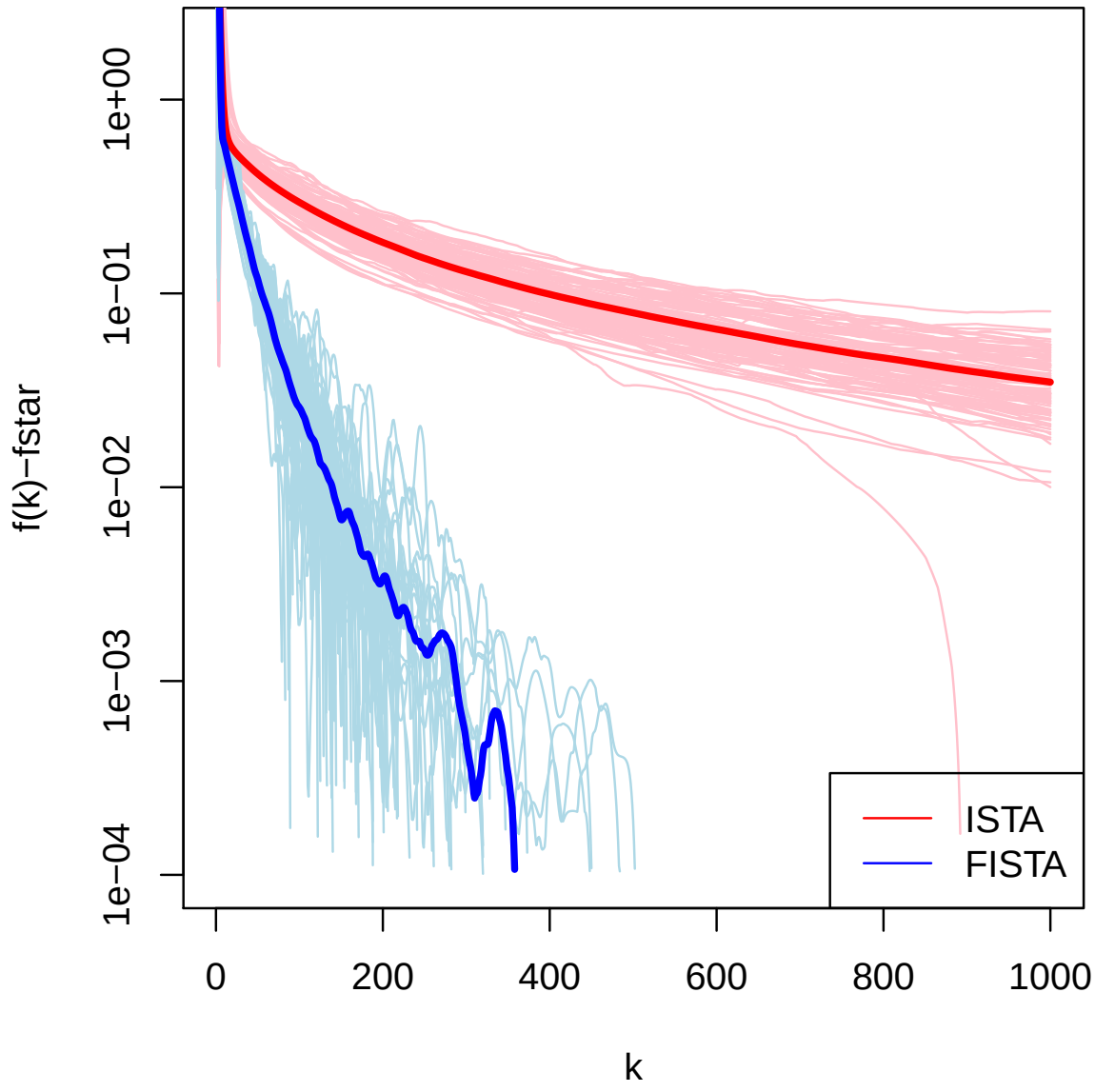
$$\beta^{(k)} = S_{\lambda t_k}(\beta^{(k-1)} + t_k X^T(y - X\beta^{(k-1)})), \quad k = 1, 2, 3, \ldots$$

$S_\lambda(\cdot)$ being vector soft-thresholding. Applying acceleration gives us FISTA (F is for Fast):[7] for $k = 1, 2, 3, \ldots$,

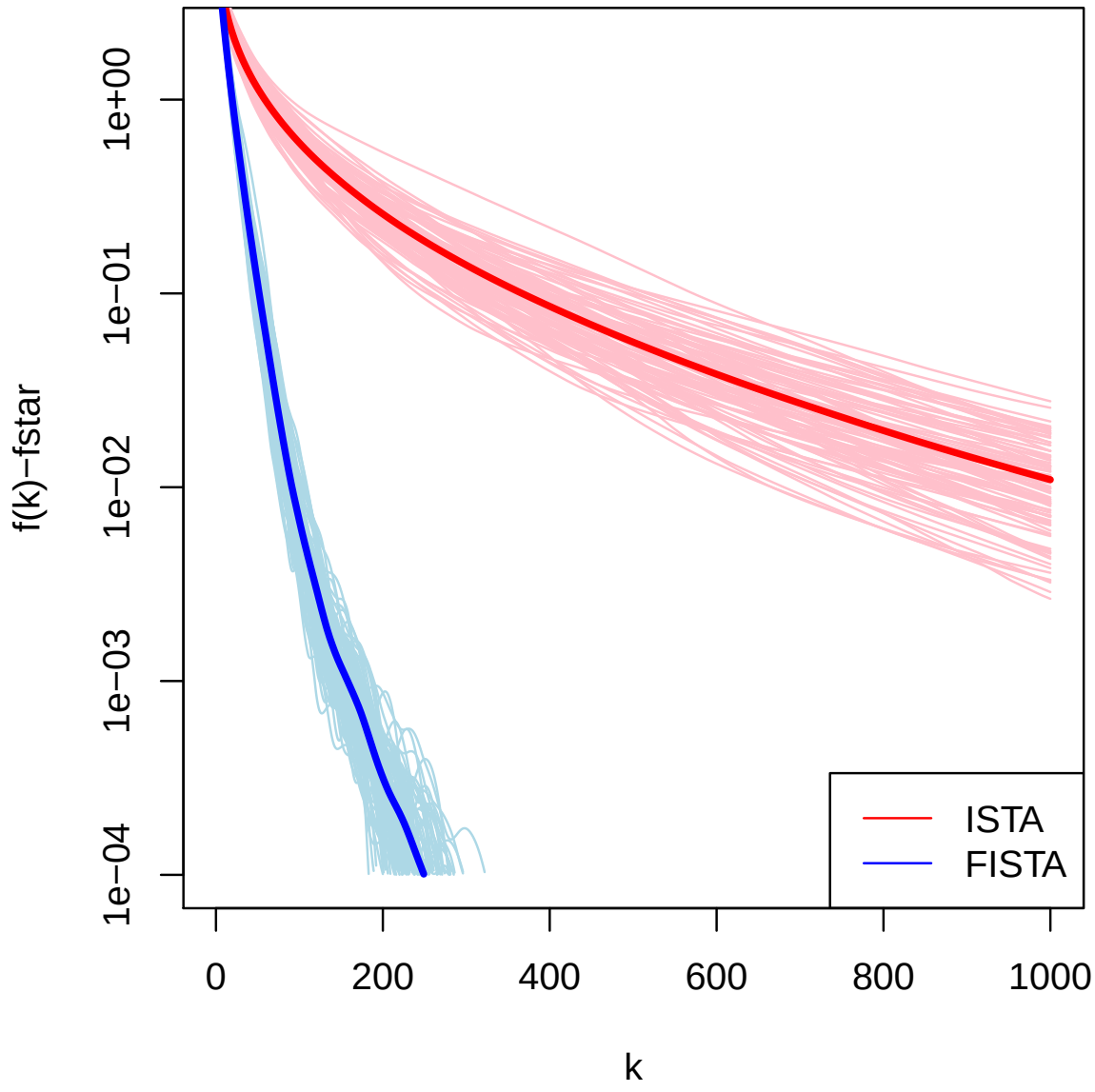
$$\begin{aligned} v &= \beta^{(k-1)} + \frac{k-2}{k+1}(\beta^{(k-1)} - \beta^{(k-2)}) \\ \beta^{(k)} &= S_{\lambda t_k}\big(v + t_k X^T(y - Xv)\big), \end{aligned}$$

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

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



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?

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

$$\lambda_1 > \lambda_2 > \ldots > \lambda_r$$

- 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

Sometimes backtracking and acceleration can be disadvantageous! Recall matrix completion problem: the proximal gradient update is

$$B^+ = S_\lambda\Big(B + t\big(P_\Omega(Y) - P^\perp(B)\big)\Big)$$

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

- One backtracking loop evaluates generalized gradient $G_t(x)$, i.e., evaluates $\mathrm{prox}_t(x)$, 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 $t = 1$),

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

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

References and further reading

- S. Boyd, Lecture notes for EE 264B, Stanford University, Spring 2010-2011
- Y. Nesterov (1998), "Introductory lectures on convex optimization: a basic course", Chapter 3
- B. Polyak (1987), "Introduction to optimization", Chapter 5
- L. Vandenberghe, Lecture notes for EE 236C, UCLA, Spring 2011-2012

References and further reading

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