Numerical Optimization

Lecture 9: Proximal Algorithms

Hao Wang
Email: wanghao1@shanghaitech.edu.cn

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

- 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:

$$x^{k+1} \leftarrow x^k - t \cdot \nabla f(x^k)$$

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

$$x^{k+1} \leftarrow \arg\min_z f(x) + \nabla f(x)^T (z - x) + \frac{1}{2t} \|z - x\|_2^2$$

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?
That is update

$$\begin{aligned} x^{k+1} \leftarrow & \arg\min g(x^k) + \nabla g(x^k)^T (z - x) + \frac{1}{2t} \|z - x^k\|_2^2 + h(z) \\ = & \arg\min \frac{1}{2t} \|z - (x^k - t\nabla g(x^k))\|_2^2 + h(z) \end{aligned}$$

- $\frac{1}{2t} \|z - (x^k - t\nabla g(x^k))\|_2^2$ stay close to gradient update for g
- $h(z)$ also make h small
- $h(z) = \Omega$ reverts to gradient descent

Let check the subproblem, define proximal mapping

$$\text{prox}_{h,t}(x) = \arg\min_z \frac{1}{2t}\|x - z\|_2^2 + h(z)$$

Proximal gradient descent: choose x^0, repeat

$$x^{k+1} \leftarrow \text{prox}_{h,t_k}(x^k - t_k \nabla g(x^k))$$

Does it work?

- $\text{prox}_{h,t}(\cdot)$ has closed-form for many important function h
- $\text{prox}_{h,t}(\cdot)$ doesn't depend on g at all, only on h
- Smooth part g can be complicated, we only need to compute its gradients
- $h(x) = 0$, $\text{prox}_{h,t}(x) = x$.

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

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

Proximal mapping is now

$$\text{prox}(\beta) = \arg\min_z \frac{1}{2t}\|\beta - z\|_2^2 + \lambda\|z\|_1 = S_{\lambda t}(\beta)$$

Here $S_\lambda(x)$ is the soft-thresholding operator,

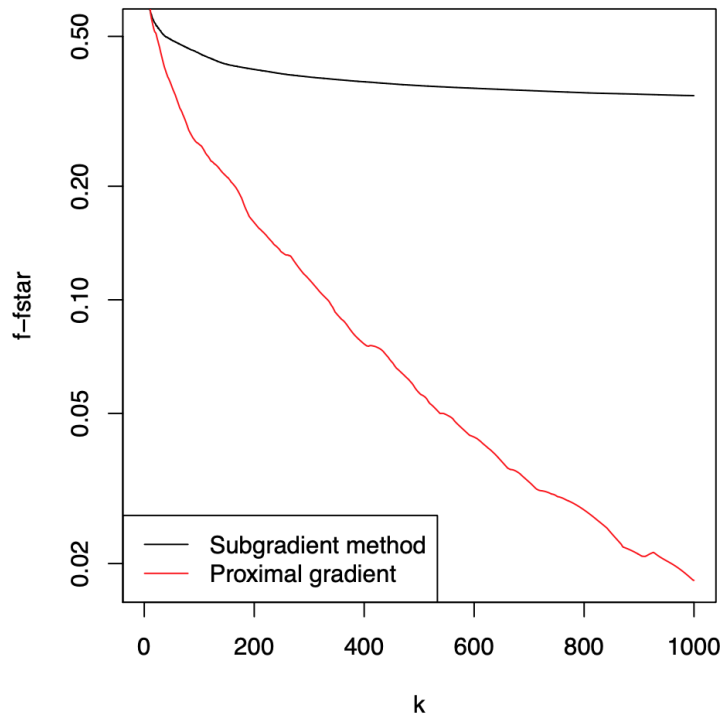
$$[S_\lambda(\beta)]_i = \begin{cases} \beta_i - \lambda & \text{if } \beta_i > \lambda \\ 0 & \text{if } -\lambda \leq \beta_i \leq \lambda, \\ \beta_i + \lambda & \text{if } \beta_i < -\lambda \end{cases}$$

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

$$\beta^{k+1} \leftarrow S_{\lambda t}(\beta^k + tX^T(y - X\beta))$$

Often called the iterative soft-thresholding algorithm (ISTA).[1]

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



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

$$x^{k+1} \leftarrow \text{prox}_{h,t_k}(x^k - t_k \nabla g(x^k))$$

What is the search direction? $x^{k+1} \leftarrow x^k - t_k G_{t_k}(x^k)$

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

Backtracking for proximal gradient descent works similar as before (in gradient descent), but operates on g, not f
Choose parameter $0\gamma < 1$. At each iteration, start at $t = t_{(0)}$ 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.

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

- g is convex, differentiable, and ∇g is Lipschitz continuous with constant $L > 0$
- h is convex, $\text{prox}_t(x) = \arg\min_z \{\|x - z\|_2^2/(2t) + h(z)\}$ can be evaluated efficiently.

Theorem 1

Proximal gradient descent with fixed stepsize $t \leq 1/L$ 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 $O(1/\epsilon)$. Matches gradient descent rate! (But remember prox cost ...)

Consider for h convex (not necessarily differentiable)

$$\min_x h(x)$$

Proximal update step is just

$$x^{k+1} = \arg\min_z \frac{1}{2t}\|x - z\|_2^2 + h(z)$$

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

Consider

$$\min_x \ f(x) = h(F(x)) + r(x)$$

Proximal update step is just

$$x^{k+1} \leftarrow \arg\min_x \ h(F(x^k) + \nabla F(x^k)(x - x^k)) + r(y) + \frac{1}{2\mu}\|y - x^k\|^2.$$

Called proximal linear algorithm. Suitable for nonlinear regression with regularization.

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

$$\text{prox}_{h,t}(x) = \arg\min_z \frac{1}{2t}\|x - z\|_2^2 + h(z)$$

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 rates[2]

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

[2] Schmidt 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 $O(1/\sqrt{\epsilon})$ convergence rate. Four ideas (three acceleration methods) by Nesterov:

- 1983: original acceleration idea for smooth functions
- another acceleration idea for smooth functions
- smoothing techniques for nonsmooth functions, coupled with original acceleration idea
- acceleration idea for composite functions (Beck and Teboulle, 2008)

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"

Consider

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

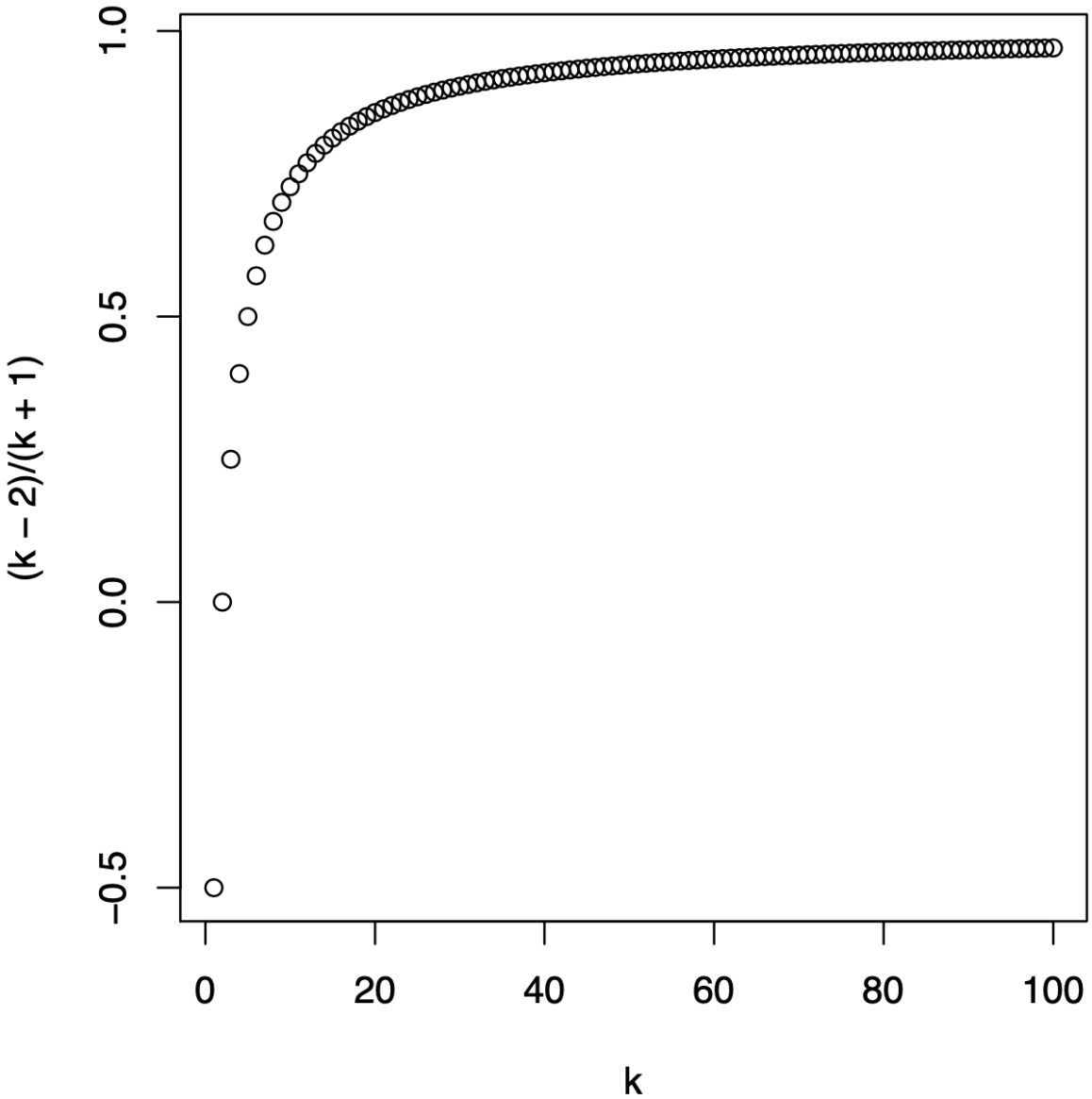
where g convex, differentiable, and h convex. Accelerated proximal gradient method: choose initial point $x^0 = x^{-1}$, repeat:

$$v = x^{k-1} + \frac{k-2}{k+1}(x^{k-1} - x^{k-2})$$
$$x^k = \text{prox}_{t_k}(v - t_k \nabla g(v))$$

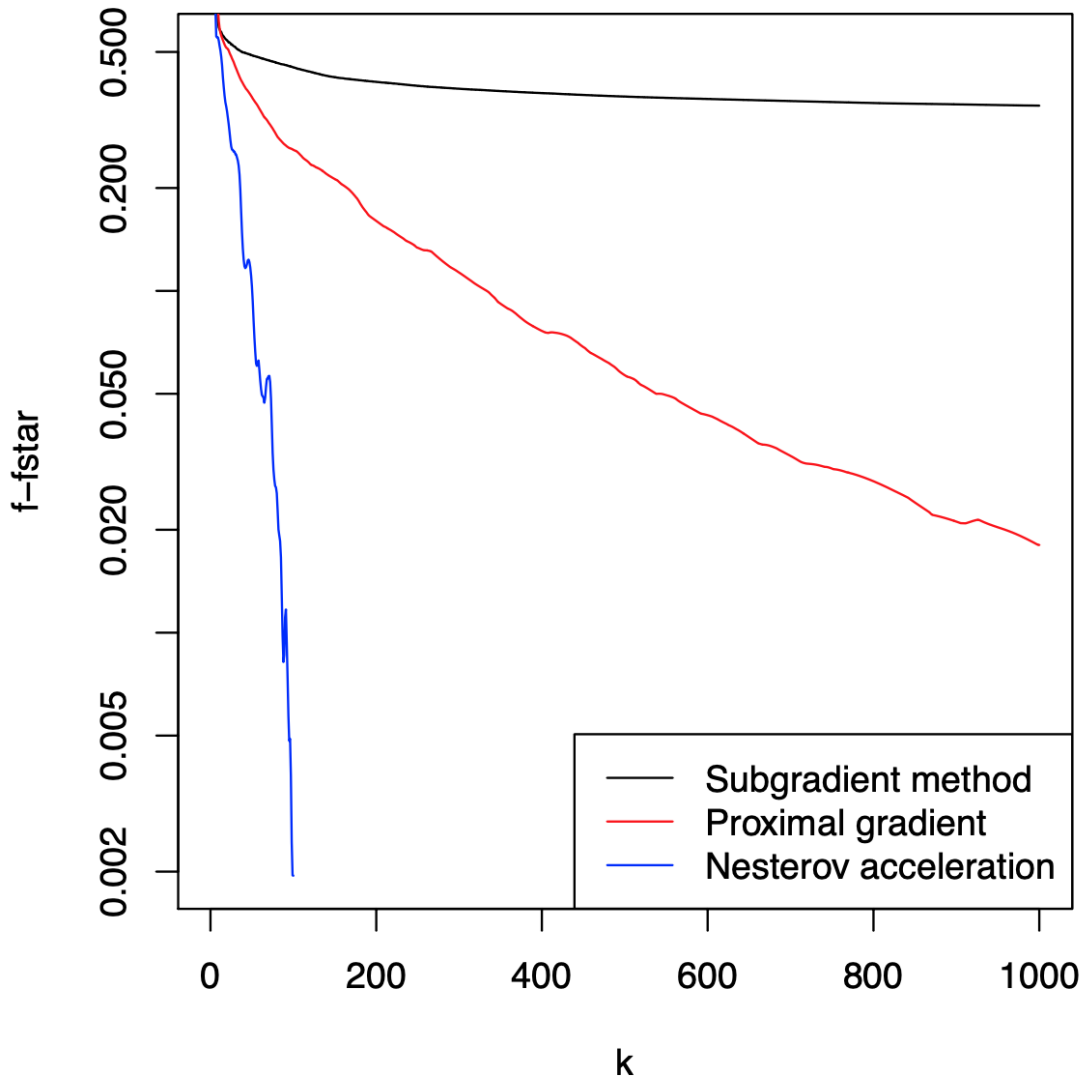
for $k = 1, 2, 3, \ldots$

- 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
- When $h = 0$ we get accelerated gradient method

Momentum weights:



Back to lasso example: acceleration can really help!



Note: accelerated proximal gradient is not a descent method

For $f = g + h$, 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, $\mathrm{prox}_t(x) = \arg\min_z \{\frac{1}{2t}\|x - z\|_2^2 + h(z)\}$ can be evaluated

Theorem 2

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 $O(1/\sqrt{\epsilon})$ for first-order methods.

Back to lasso problem:

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

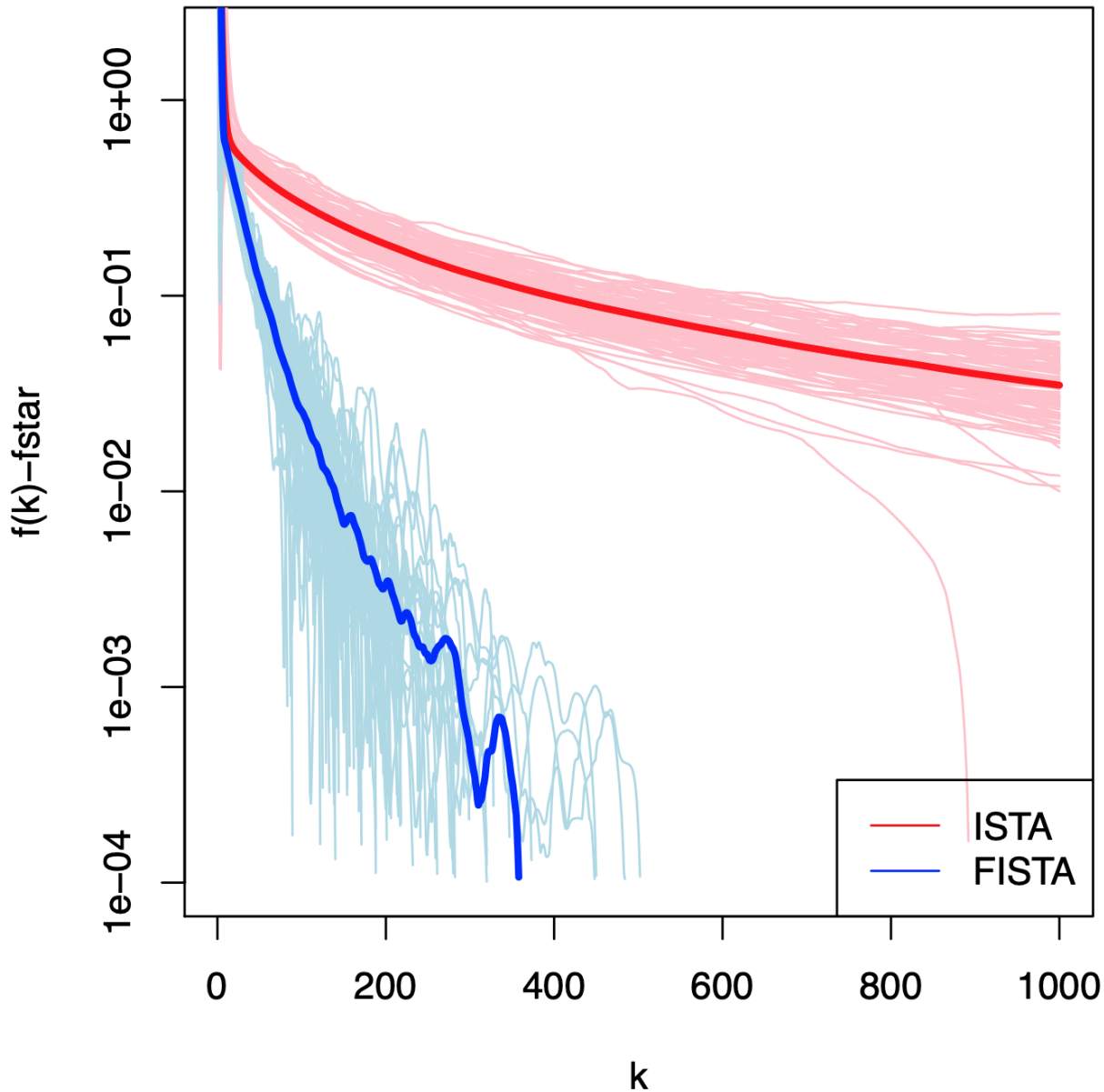
Recall ISTA (Iterative Soft-thresholding Algorithm):

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

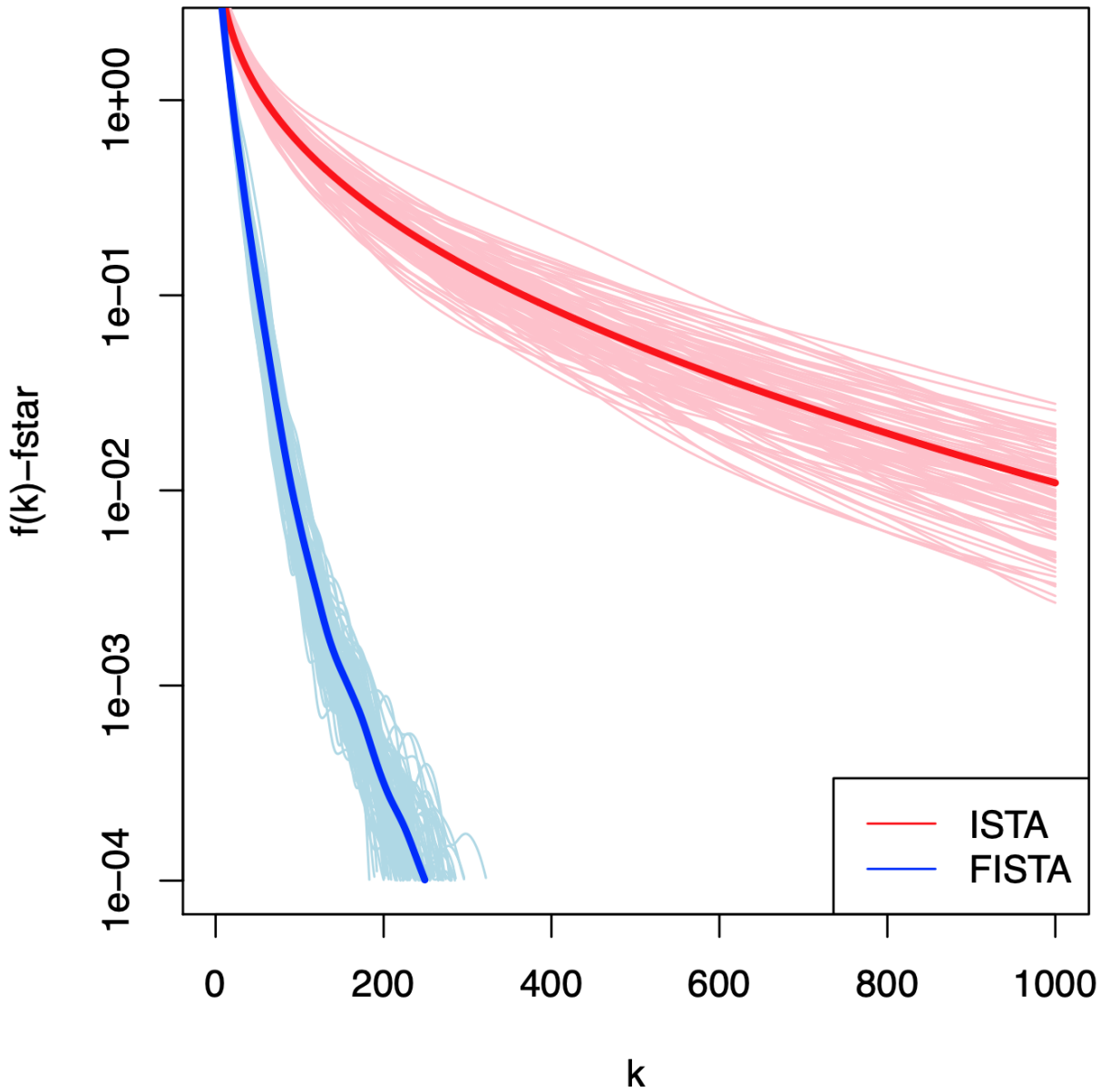
$S_\lambda(\cdot)$ being vector soft-thresholding. Applying acceleration gives us FISTA (F is for Fast): for $k = 1, 2, 3, ...$

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

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



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



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.
For example, 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