Acceleration

Barnabas Poczos & Ryan Tibshirani
Convex Optimization 10-725/36-725

Recall generalized gradient descent

We want to solve

$$\min_{x \in \mathbb{R}^n} \; g(x) + h(x),$$

for g convex and differentiable, h convex

Generalized gradient descent: choose initial $x^{(0)} \in \mathbb{R}^n$, repeat:

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

where the prox function is defined as

$$\mathrm{prox}_t(x) = \operatorname*{argmin}_{z \in \mathbb{R}^n} \; \frac{1}{2t}\|x - z\|^2 + h(z)$$

If ∇g is Lipschitz continuous, and prox function can be evaluated, then generalized gradient has rate $O(1/k)$ (counts # of iterations)

We can apply acceleration to achieve optimal $O(1/k^2)$ rate!

Acceleration

Turns out we can accelerate generalized gradient descent in order to achieve the optimal $O(1/k^2)$ convergence rate

Four ideas (three acceleration methods) by Nesterov:

- 1983: original accleration 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[1]

Beck and Teboulle (2008): extension of Nesterov (1983) to composite functions[2]

Tseng (2008): unified analysis of accleration techniques (all of these, and more)

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

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

Outline

Rest of today:

- Acceleration for composite functions (method of Beck and Teboulle (2008), presentation of Vandenberghe's notes)
- Convergence rate
- FISTA
- Is acceleration always useful?

Accelerated generalized gradient method

Our problem

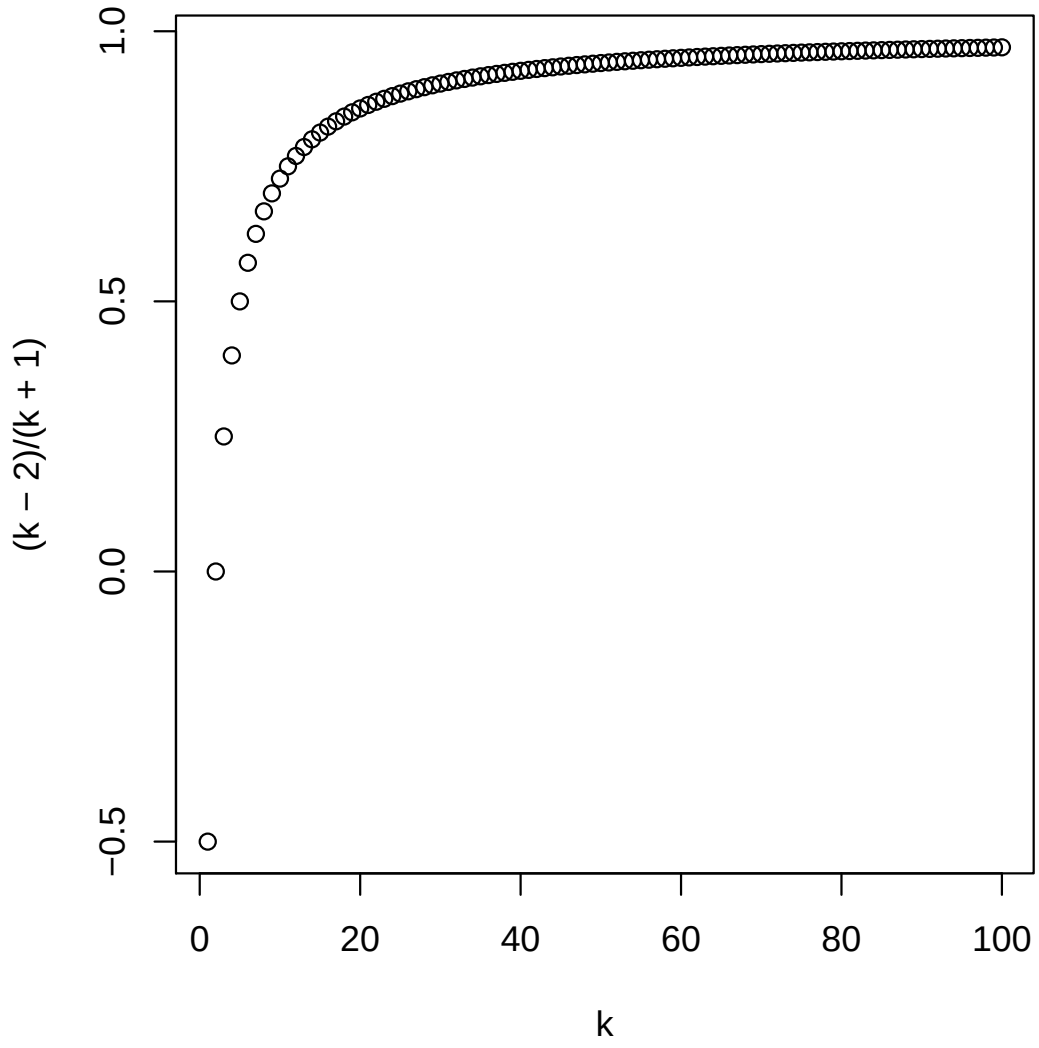
$$\min_{x \in \mathbb{R}^n} \; g(x) + h(x),$$

for g convex and differentiable, h convex

Accelerated generalized gradient method: choose an initial point $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$, repeat for $k = 1, 2, 3, \ldots$

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

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


1.0
0.5
0.0
−0.5
(k − 2)/(k + 1)
0
20
40
60
80
100
k

Consider minimizing

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

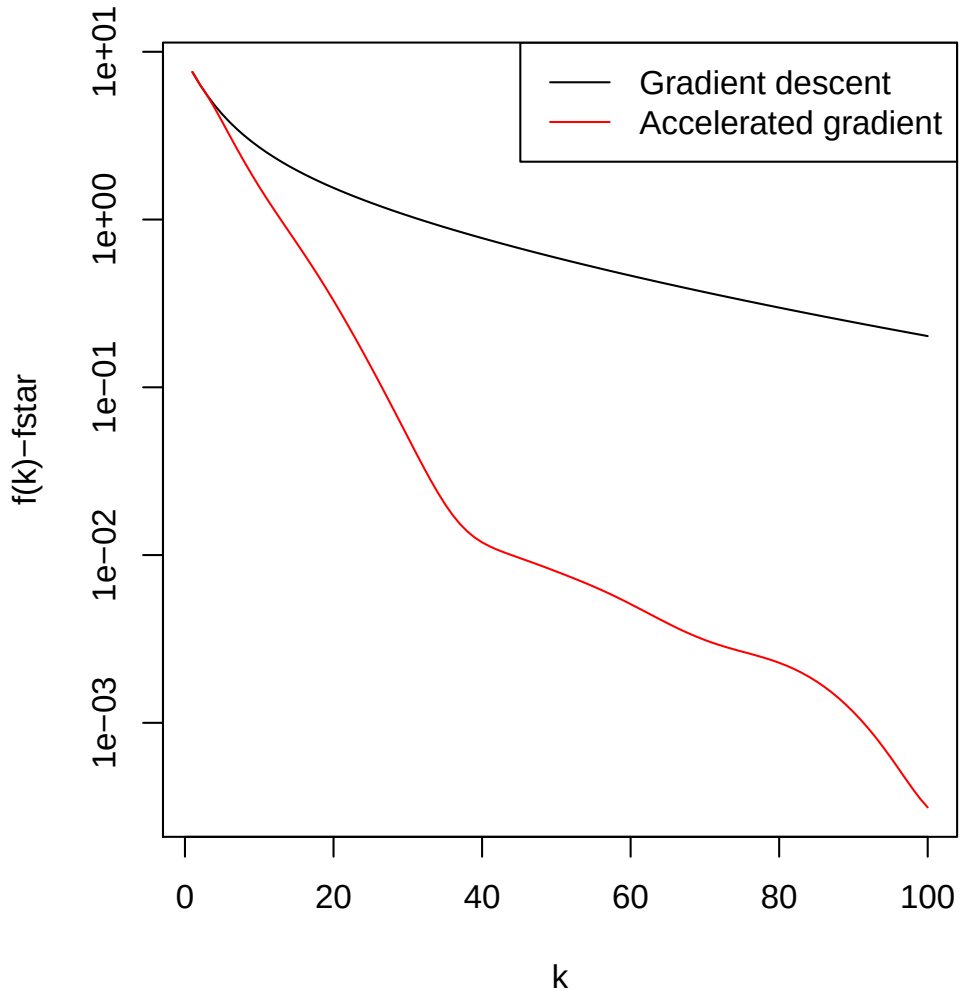
i.e., logistic regression with predictors $x_i \in \mathbb{R}^p$

This is smooth, and

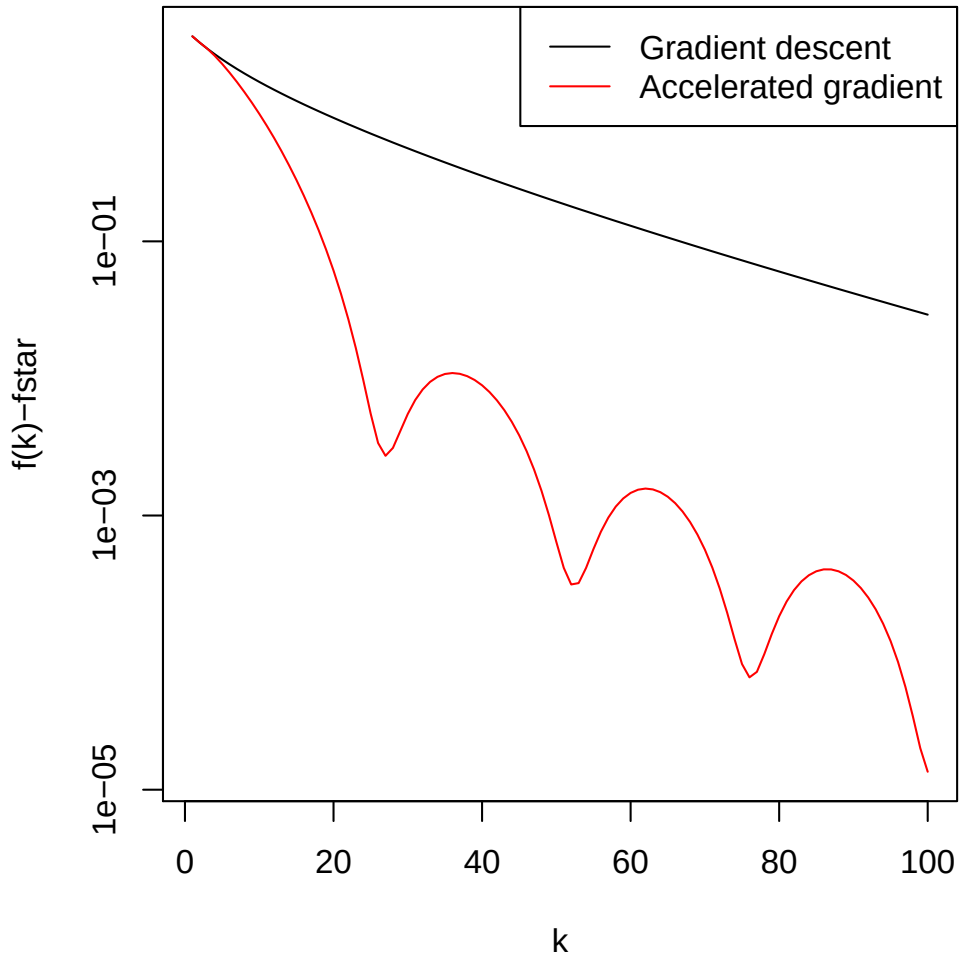
$$\begin{aligned} \nabla f(\beta) &= -X^T(y - p(\beta)), \quad \text{where} \\ p_i(\beta) &= \exp(x_i^T \beta)/(1 + \exp(x_i^T \beta)) \quad \text{for } i = 1, \ldots n \end{aligned}$$

No nonsmooth part here, so $\text{prox}_t(\beta) = \beta$

Example (with $n = 30$, $p = 10$):



Another example ($n = 30$, $p = 10$):



Not a descent method!

Reformulation

Initialize $x^{(0)} = u^{(0)}$, and repeat for $k = 1, 2, 3, \ldots$

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

with $\theta_k = 2/(k+1)$

This is equivalent to the formulation of accelerated generalized gradient method presented earlier (slide 5). Makes convergence analysis easier

(Note: Beck and Teboulle (2008) use a choice $\theta_k < 2/(k+1)$, but very close)

Convergence analysis

As usual, we are minimizing $f(x) = g(x) + h(x)$ assuming

- g is convex, differentiable, ∇g is Lipschitz continuous with constant $L > 0$
- h is convex, prox function can be evaluated

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

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

Achieves the optimal $O(1/k^2)$ rate for first-order methods!

I.e., to get $f(x^{(k)}) - f(x^\star) \leq \epsilon$, need $O(1/\sqrt{\epsilon})$ iterations

Helpful inequalities

We will use

$$\frac{1-\theta_k}{\theta_k^2} \leq \frac{1}{\theta_{k-1}^2}, \quad k = 1, 2, 3, \ldots$$

We will also use

$$h(v) \leq h(z) + \frac{1}{t}(v-w)^T(z-v), \quad \text{all } z, w, v = \text{prox}_t(w)$$

Why is this true? By definition of prox operator,

$$\begin{aligned} v \text{ minimizes } \frac{1}{2t}\|w-v\|_2^2 + h(v) &\iff 0 \in \frac{1}{t}(v-w) + \partial h(v) \\ &\iff -\frac{1}{t}(v-w) \in \partial h(v) \end{aligned}$$

Now apply definition of subgradient

Convergence proof

Key steps:

- g Lipschitz with constant $L > 0$ and $t \leq 1/L \Rightarrow$

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

- From our bound using prox operator,

$$h(x^+) \leq h(z) + \frac{1}{t}(x^+ - y)^T(z - x^+) + \nabla g(y)^T(z - x^+) \quad \text{all } z$$

- Adding these together and using convexity of g,

$$f(x^+) \leq f(z) + \frac{1}{t}(x^+ - y)^T(z - x^+) + \frac{1}{2t}\|x^+ - y\|_2^2 \quad \text{all } z$$

- Using this bound at $z = x$ and $z = x^*$:

$$\begin{aligned} & f(x^+) - f(x^\star) - (1-\theta)(f(x) - f(x^\star)) \\ & \quad \leq \frac{1}{t}(x^+ - y)^T(\theta x^\star + (1-\theta)x - x^+) + \frac{1}{2t}\|x^+ - y\|_2^2 \\ & \quad = \frac{\theta^2}{2t}\Big(\|u - x^\star\|_2^2 - \|u^+ - x^\star\|_2^2\Big) \end{aligned}$$

- I.e., at iteration k,

$$\begin{aligned} & \frac{t}{\theta_k^2}(f(x^{(k)}) - f(x^\star)) + \frac{1}{2}\|u^{(k)} - x^\star\|_2^2 \\ & \quad \leq \frac{(1-\theta_k)t}{\theta_k^2}(f(x^{(k-1)}) - f(x^\star)) + \frac{1}{2}\|u^{(k-1)} - x^\star\|_2^2 \end{aligned}$$

- Using $(1-\theta_i)/\theta_i^2 \leq 1/\theta_{i-1}^2$, and iterating this inequality,

$$\begin{aligned}
&\frac{t}{\theta_k^2}(f(x^{(k)}) - f(x^\star)) + \frac{1}{2}\|u^{(k)} - x^\star\|_2^2 \\
&\quad \leq \frac{(1-\theta_1)t}{\theta_1^2}(f(x^{(0)}) - f(x^\star)) + \frac{1}{2}\|u^{(0)} - x^\star\|_2^2 \\
&\quad = \frac{1}{2}\|x^{(0)} - x^\star\|_2^2
\end{aligned}$$

- Therefore

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

□

Backtracking line search

A few ways to do this with acceleration ... here's a simple method (more complicated strategies exist)

First think: what do we need t to satisfy? Looking back at proof with $t_k = t \leq 1/L$,

- We used

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

- We also used

$$\frac{(1-\theta_k)t_k}{\theta_k^2} \leq \frac{t_{k-1}}{\theta_{k-1}^2},$$

  so it suffices to have $t_k \leq t_{k-1}$, i.e., decreasing step sizes

Backtracking algorithm: fix $\beta < 1$, $t_0 = 1$. At iteration k, replace x update (i.e., computation of x^+) with:

- Start with $t_k = t_{k-1}$ and $x^+ = \mathrm{prox}_{t_k}(y - t_k \nabla g(y))$
- While $g(x^+) > g(y) + \nabla g(y)^T (x^+ - y) + \frac{1}{2t_k}\|x^+ - y\|_2^2$, repeat:
  - $t_k = \beta t_k$ and $x^+ = \mathrm{prox}_{t_k}(y - t_k \nabla g(y))$

Note this achieves both requirements. So under same conditions (∇g Lipschitz, prox function evaluable), we get same rate

Theorem: Accelerated generalized gradient method with backtracking line search satisfies

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

where $t_{\mathsf{min}} = \min\{1, \beta/L\}$

FISTA

Recall lasso problem,

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

and ISTA (Iterative Soft-thresholding Algorithm):

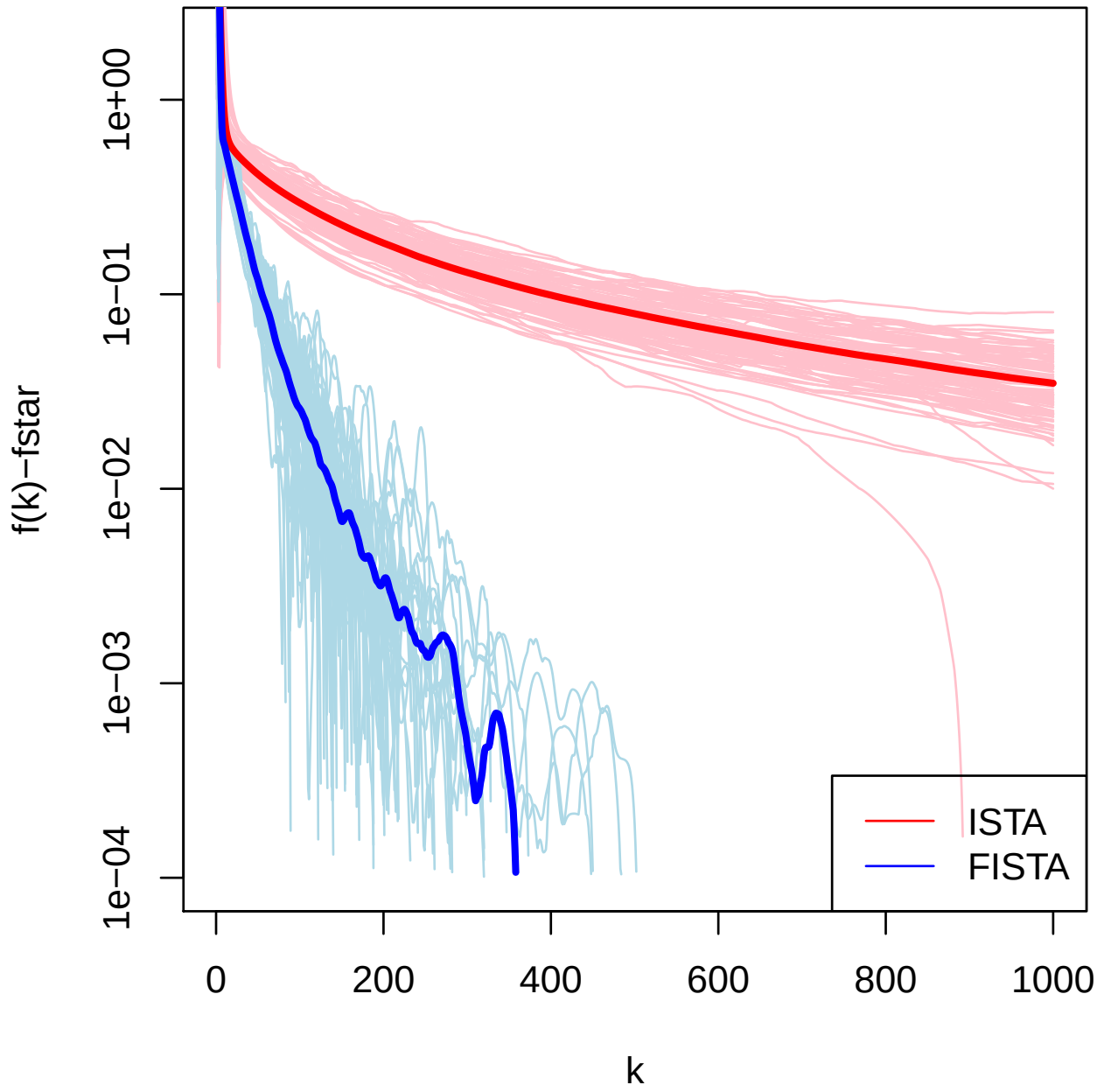
$$\beta^{(k)} = S_{\lambda t_k}\big(\beta^{(k-1)} + t_k X^T(y - X\beta^{(k-1)})\big), \quad k = 1, 2, 3, \ldots$$

$S_\lambda(\cdot)$ being vector soft-thresholding. Applying acceleration gives us FISTA (F is for Fast):[3]

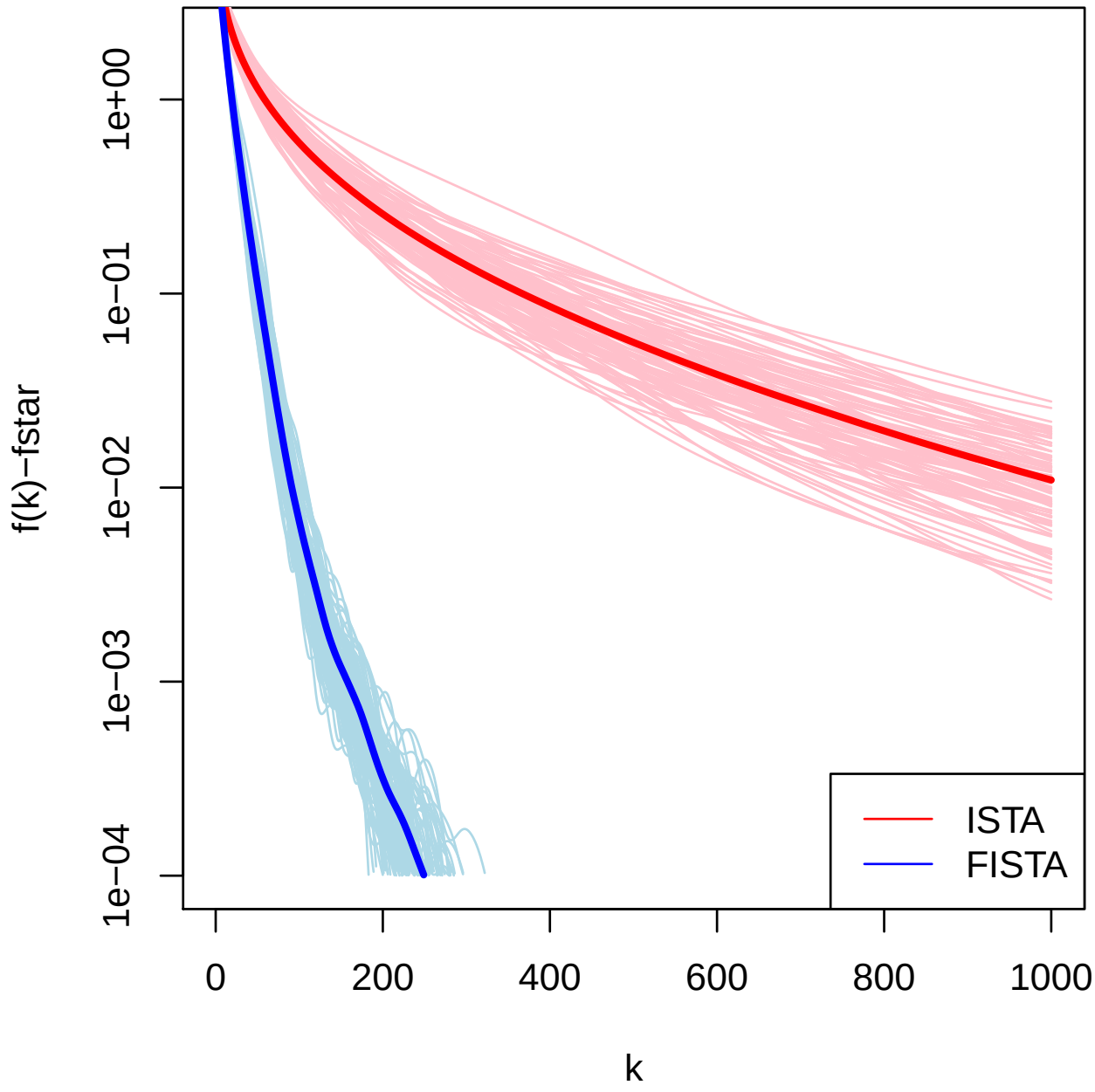
$$\begin{aligned} v &= \beta^{(k-1)} + \frac{k-2}{k+1}(\beta^{(k-1)} - \beta^{(k-2)}) \\ \beta^{(k)} &= S_{\lambda t_k}\big(v + t_k X^T(y - Xv)\big), \qquad k = 1, 2, 3, \ldots \end{aligned}$$

[3] 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 is generally 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. I.e., 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, generalized gradient descent can often perform just as well without acceleration

Sometimes backtracking and acceleration can be disadvantageous!

Recall matrix completion problem: observe some only entries of A, $(i,j) \in \Omega$, we want to fill in the rest, so we solve

$$\min_{B \in \mathbb{R}^{m \times n}} \frac{1}{2}\|P_\Omega(A) - P_\Omega(B)\|_F^2 + \lambda \|B\|_*$$

where $\|B\|_* = \sum_{i=1}^r \sigma_i(B)$, nuclear norm, and

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

Generalized gradient descent with $t = 1$ (soft-impute algorithm): updates are

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

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

Backtracking line search with generalized gradient:

- Each backtracking loop evaluates generalized gradient $G_t(x)$ at various values of t
- Hence requires multiple evaluations of $\mathrm{prox}_t(x)$
- For matrix completion, can't afford this!

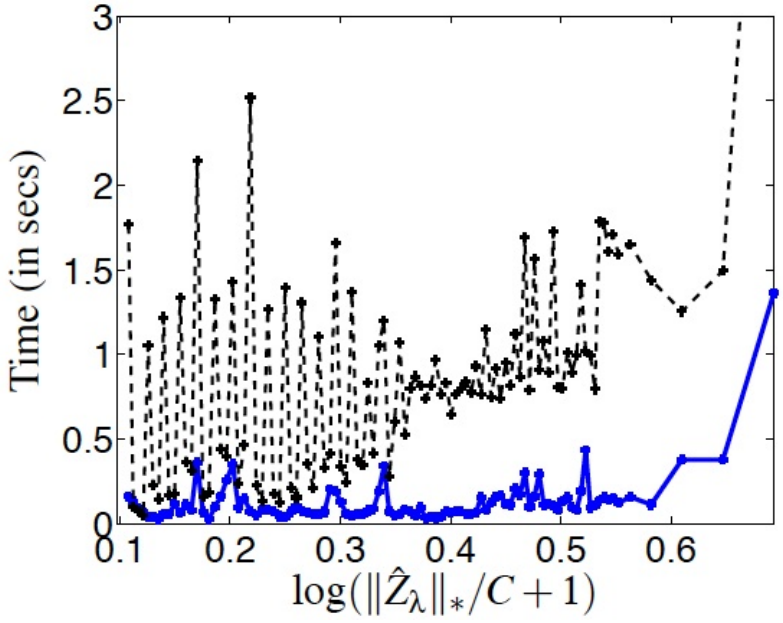
Acceleration with generalized gradient:

- Changes argument we pass to prox function: $y - t\nabla g(y)$ instead of $x - t\nabla g(x)$
- For matrix completion (and $t = 1$),

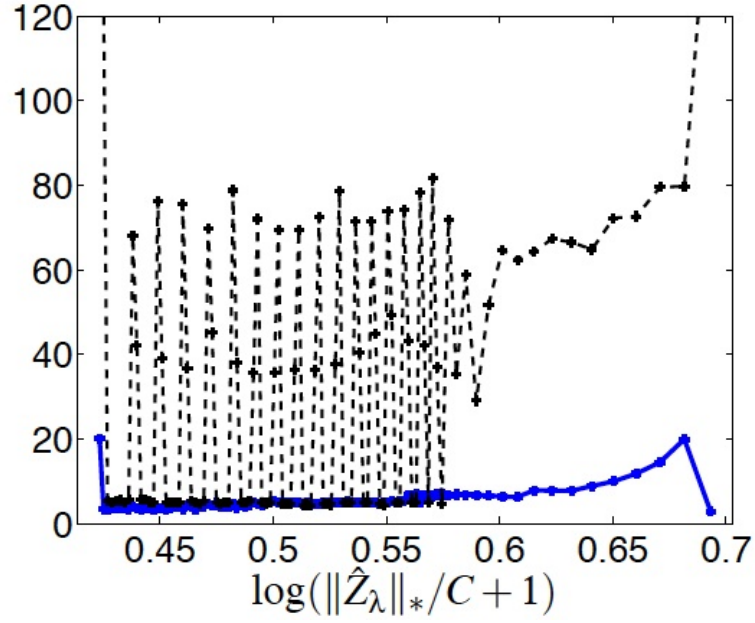
$$B - \nabla g(B) = \underbrace{P_\Omega(A)}_{\text{sparse}} + \underbrace{P_\Omega^\perp(B)}_{\text{low rank}} \quad \Rightarrow \quad \text{fast SVD}$$

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

Soft-impute uses $L = 1$ and exploits special structure ... so it can outperform fancier methods. E.g., soft-impute (solid blue line) vs accelerated generalized gradient (dashed black line):



Small problem



Big problem

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

References

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"

and there are many more ...

Helpful lecture notes/books:

- E. Candes, Lecture Notes for Math 301, Stanford University, Winter 2010-2011
- Y. Nesterov (2004), "Introductory lectures on convex optimization: a basic course", Chapter 2
- L. Vandenberghe, Lecture Notes for EE 236C, UCLA, Spring 2011-2012