Proximal Gradient Descent

Lecturer: Aarti Singh
Co-instructor: Pradeep Ravikumar

Convex Optimization 10-725/36-725

Outline

Today:

- Proximal gradient descent
- Subgradients
- Convergence analysis
- ISTA, matrix completion
- Accelerated proximal gradient descent, FISTA

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,

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

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. Keep in mind that each iteration evaluates $\mathrm{prox}_t(\cdot)$ once, and this can be cheap or expensive, depending on h

Evaluating the prox operator

Proximal mapping:

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

How to evaluate the prox operator? Since h is not differentiable,

we need to notion of sub-gradients.

Subgradients

Recall that for convex and differentiable f,

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{for all } x, y$$

I.e., linear approximation always underestimates f

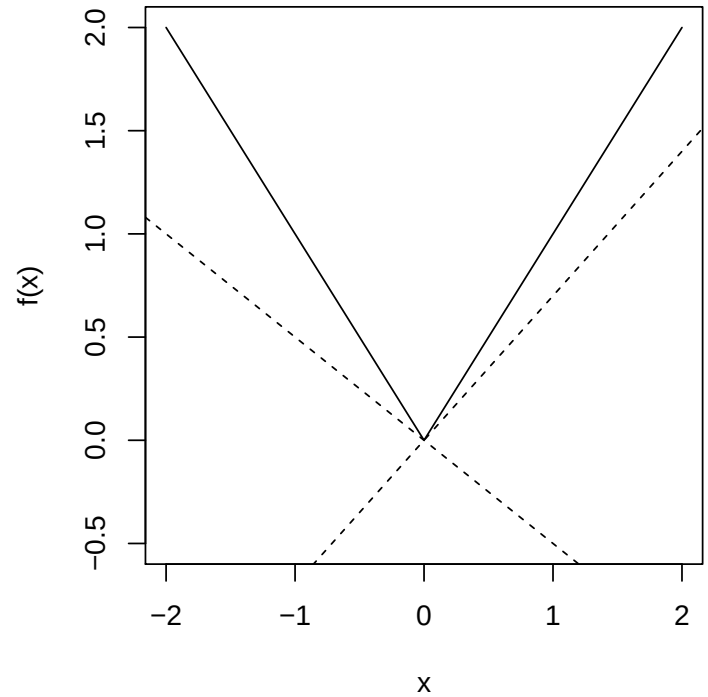
A subgradient of a convex function f at x is any $g \in \mathbb{R}^n$ such that

$$f(y) \geq f(x) + g^T (y - x) \quad \text{for all } y$$

- Always exists
- If f differentiable at x, then $g = \nabla f(x)$ uniquely
- Actually, same definition works for nonconvex f (however, subgradients need not exist)

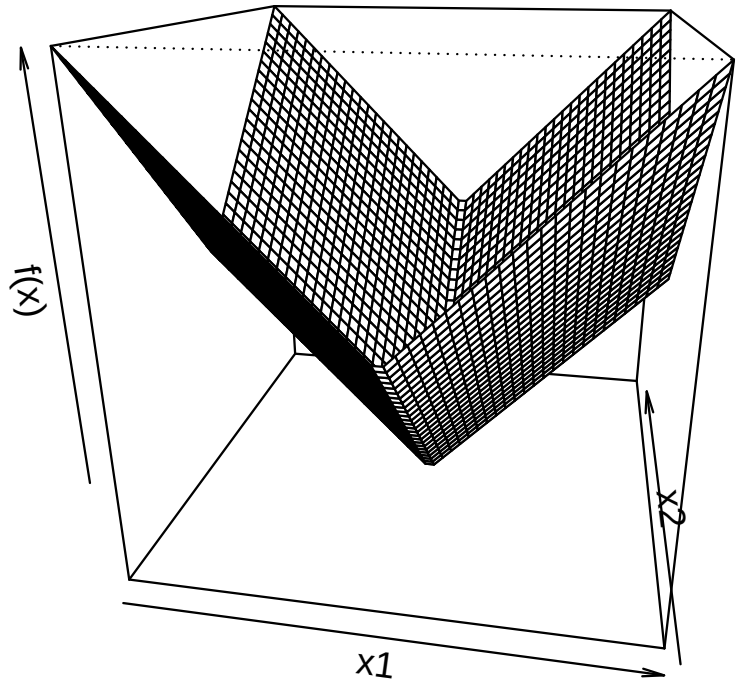
Examples of subgradients

Consider $f : \mathbb{R} \to \mathbb{R}$, $f(x) = |x|$



- For $x \neq 0$, unique subgradient $g = \mathrm{sign}(x)$
- For $x = 0$, subgradient g is any element of $[-1, 1]$

Consider $f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = \|x\|_1$



- For $x_i \neq 0$, unique ith component $g_i = \text{sign}(x_i)$
- For $x_i = 0$, ith component g_i is any element of $[-1, 1]$

Optimality condition for unconstrained problems

Set of all subgradients of convex f is called the subdifferential:

$$\partial f(x) = \{g \in \mathbb{R}^n : g \text{ is a subgradient of } f \text{ at } x\}$$

For any f (convex or not),

$$f(x^\star) = \min_x \; f(x) \quad \Longleftrightarrow \quad 0 \in \partial f(x^\star)$$

I.e., $x^\star$ is a minimizer if and only if 0 is a subgradient of f at $x^\star$. This is called the subgradient optimality condition

Why? Easy: $g = 0$ being a subgradient means that for all y

$$f(y) \geq f(x^\star) + 0^T(y - x^\star) = f(x^\star)$$

Note the implication for a convex and differentiable function f, with $\partial f(x) = \{\nabla f(x)\}$

Example: soft-thresholding

Simplfied lasso problem with $X = I$:

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

Subgradient optimality:

$$\begin{aligned} 0 \in \partial\Big(\frac{1}{2}\|y - \beta\|_2^2 + \lambda\|\beta\|_1\Big) & \\ \iff & \ 0 \in -(y - \beta) + \lambda\partial\|\beta\|_1 \\ \iff & \ (y - \beta) = \lambda v \end{aligned}$$

for some $v \in \partial\|\beta\|_1$, i.e.,

$$v_i \in \begin{cases} \{1\} & \text{if } \beta_i > 0 \\ \{-1\} & \text{if } \beta_i < 0 \\ [-1, 1] & \text{if } \beta_i = 0 \end{cases}, \quad i = 1, \ldots p$$

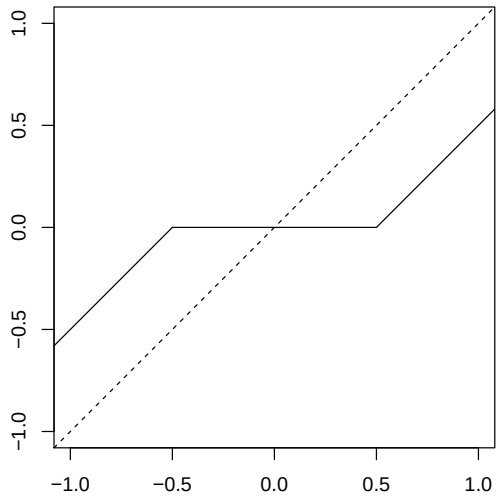
From last slide, subgradient optimality conditions are

$$\begin{cases} y_i - \beta_i = \lambda \cdot \text{sign}(\beta_i) & \text{if } \beta_i \neq 0 \\ |y_i - \beta_i| \leq \lambda & \text{if } \beta_i = 0 \end{cases}$$

Therefore, solution is $\beta = S_\lambda(y)$, where S_λ is the soft-thresholding operator:

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

Soft-thresholding in one variable:



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,

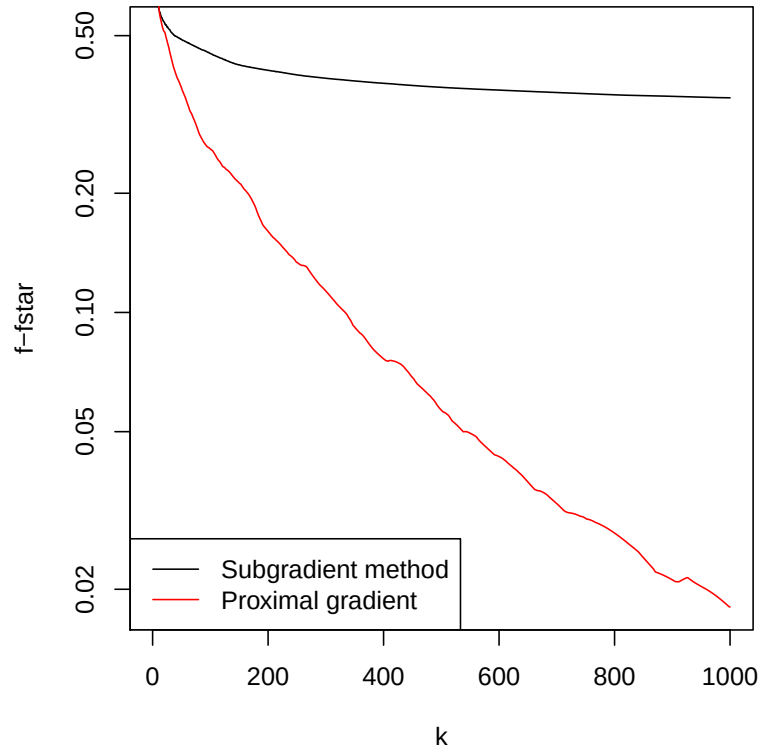
$$[S_\lambda(\beta)]_i = \begin{cases} \beta_i - \lambda & \text{if } \beta_i > \lambda \\ 0 & \text{if } -\lambda \leq \beta_i \leq \lambda \\ \beta_i + \lambda & \text{if } \beta_i < -\lambda \end{cases}, \quad i = 1, \ldots n$$

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).[1] Very simple algorithm

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



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

Convergence analysis

With 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

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

Proximal gradient descent has convergence rate $O(1/k)$, or $O(1/\epsilon)$

Same as gradient descent! But remember, this counts the number of iterations, not operations

Backtracking line search

Similar to gradient descent, but operates on g and not f. We fix a parameter $0 < \beta < 1$. At each iteration, start with $t = 1$, 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 $t = \beta t$. Else perform prox gradient update

Under same assumptions, we get the same rate

Theorem: Proximal gradient descent with backtracking line search satisfies

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

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

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

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

Why? Note that $\mathrm{prox}_t(B) = Z$, where Z satisfies

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

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

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

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[2], simple and effective method for matrix completion

[2] 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 $f = g + h$:

- $h = 0 \rightarrow$ gradient descent
- $h = I_C \rightarrow$ projected gradient descent
- $g = 0 \rightarrow$ proximal minimization 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

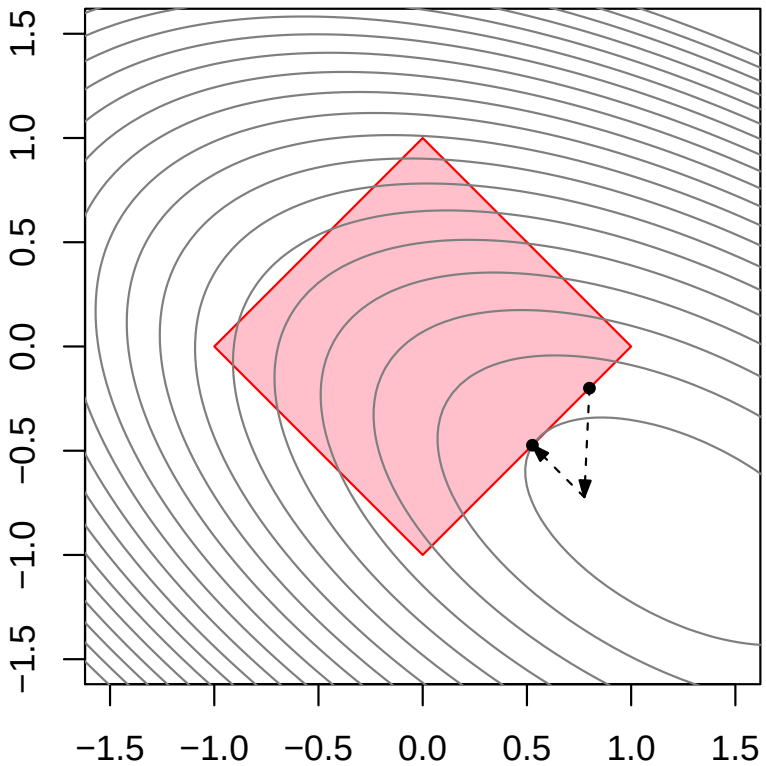
$$\begin{aligned} \text{prox}_t(x) &= \underset{z}{\text{argmin}} \; \frac{1}{2t}\|x - z\|_2^2 + I_C(z) \\ &= \underset{z \in C}{\text{argmin}} \; \|x - z\|_2^2 \end{aligned}$$

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 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

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

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[3]

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

[3] 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[4]

We will follow Beck and Teboulle (2008), extension of Nesterov (1983) to composite functions[5]

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

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

Accelerated proximal gradient method

Our problem, as before:

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

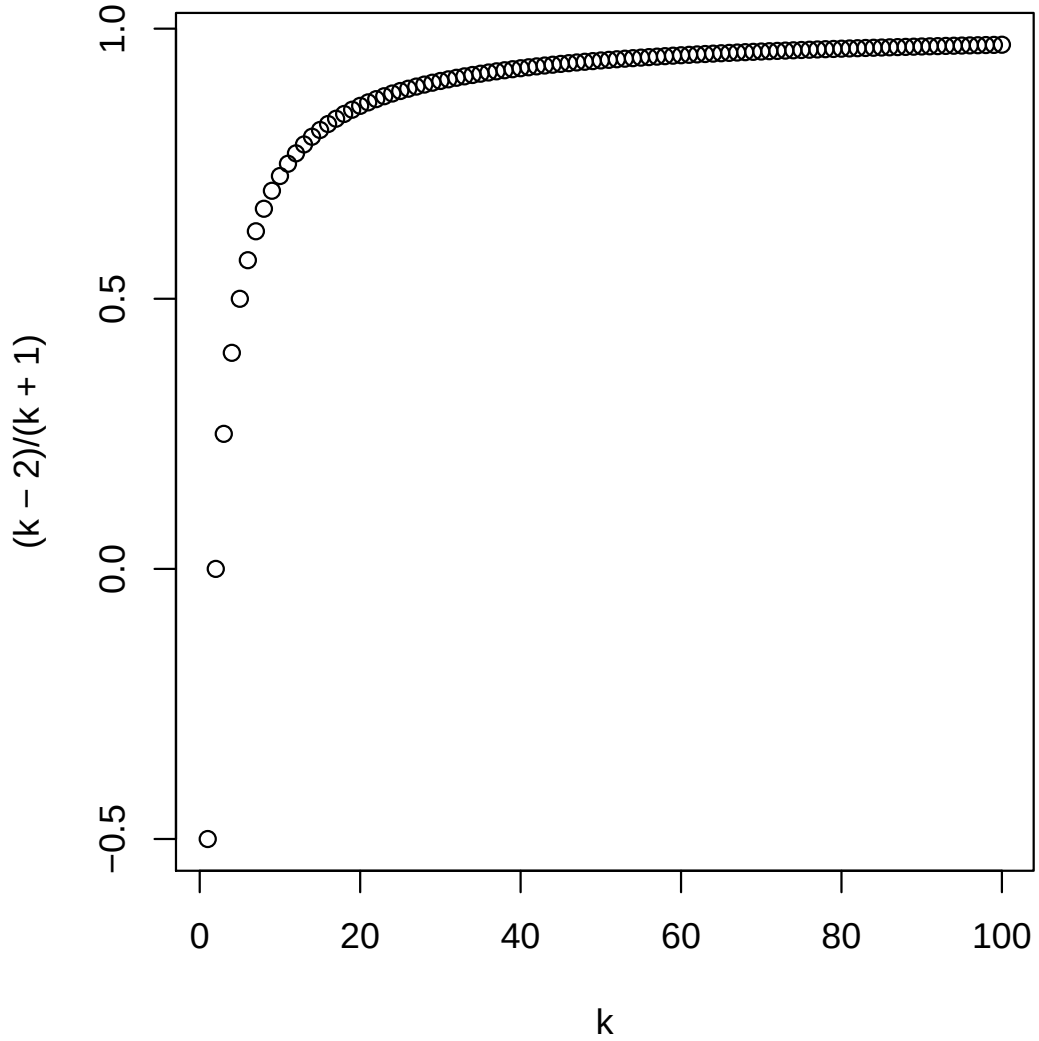
where g convex, differentiable, and h convex. Accelerated proximal gradient method: choose initial point $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$, repeat:

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

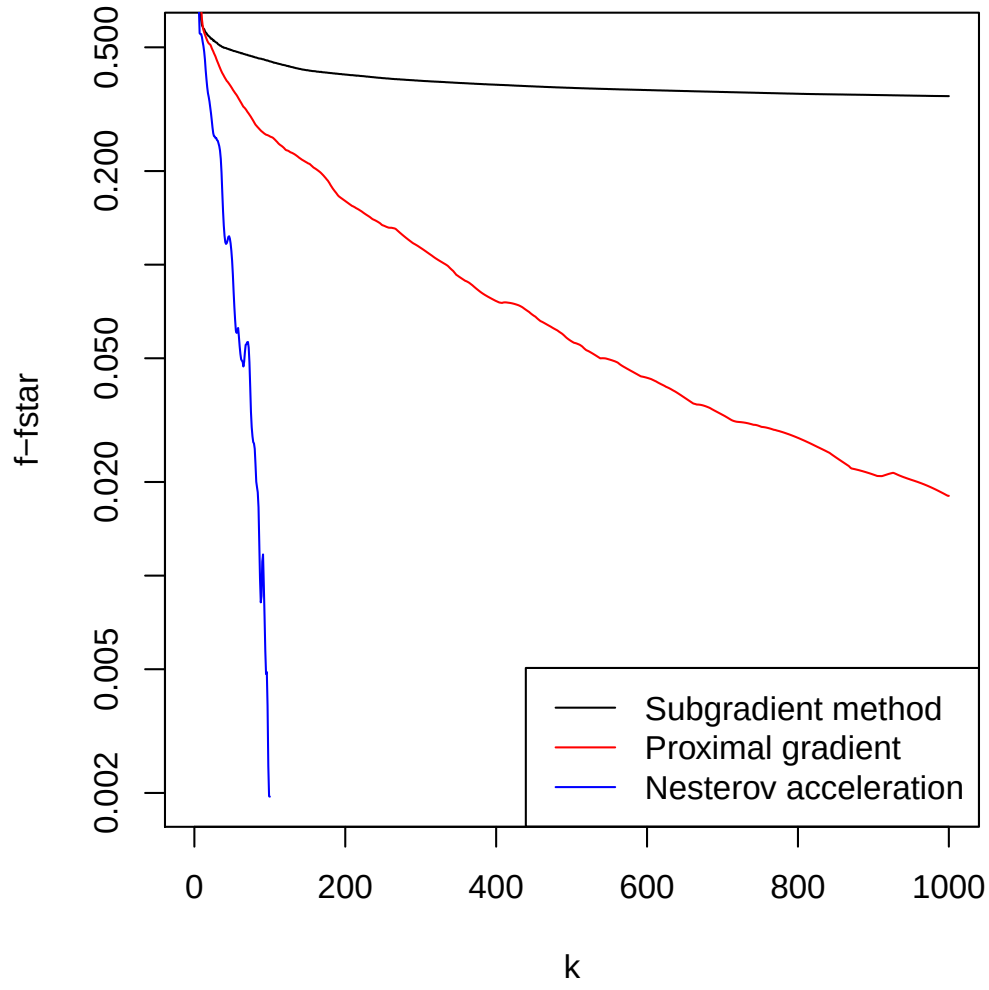
for $k = 1, 2, 3, \ldots$

- 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 ("Nesterov ripples")

Convergence analysis

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

- g is convex, differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$, and ∇g is Lipschitz continuous with constant $L > 0$
- h is convex, prox function can be evaluated

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

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

Achieves the optimal rate $O(1/k^2)$ for first-order methods! I.e., a rate of $O(1/\sqrt{\epsilon})$

Backtracking line search

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

Under same assumptions, we get the same rate

Theorem: Accelerated proximal gradient method with backtracking line search satisfies

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

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):

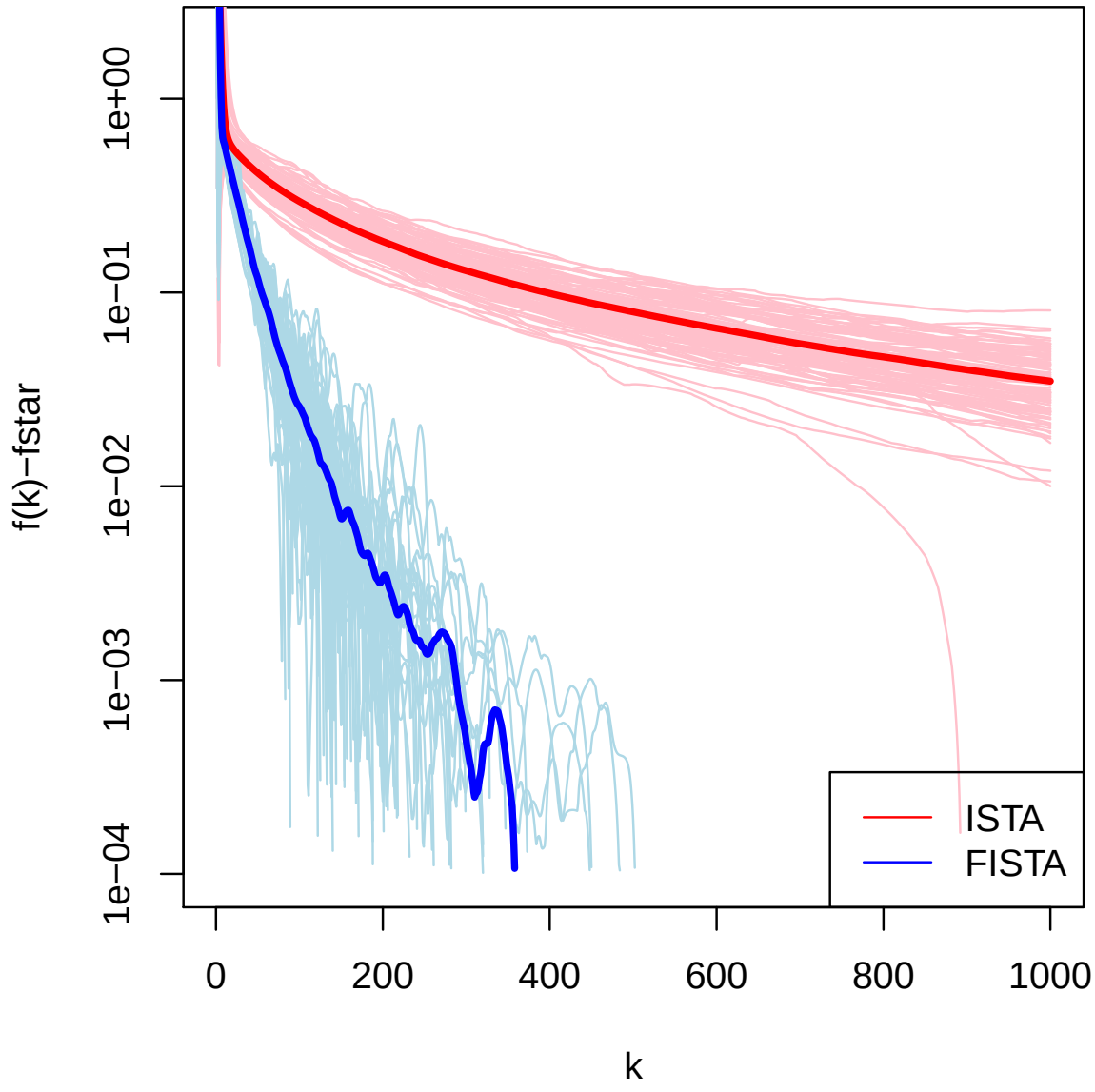
$$\beta^{(k)} = S_{\lambda t_k}\big(\beta^{(k-1)} + t_k X^T(y - X\beta^{(k-1)})\big), \quad k = 1, 2, 3, \ldots$$

$S_\lambda(\cdot)$ being vector soft-thresholding. Applying acceleration gives us FISTA (F is for Fast):[6] for $k = 1, 2, 3, \ldots$,

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

[6] 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 ($n = 100$, $p = 500$):