# Proximal Gradient Descent

## (and Acceleration)

Ryan Tibshirani
Convex Optimization 10-725

# Last time: subgradient method

Consider the problem

$$\min_x \; f(x)$$

with $f$ convex, and $\mathrm{dom}(f) = \mathbb{R}^n$. Subgradient method: choose an initial $x^{(0)} \in \mathbb{R}^n$, and repeat:

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

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

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

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

# Outline

Today:

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

# Composite 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 $\frac{1}{t} I$,

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

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?

That is, update

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

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

# Proximal gradient descent

Define proximal mapping:

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

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

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

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 $f$,

$$G_t(x) = \frac{x - \operatorname{prox}_{h,t}\big(x - t\nabla g(x)\big)}{t}$$

# 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 $\mathrm{prox}_{h,t}(\cdot)$ has a closed-form for many important functions $h$. Note:

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

Convergence analysis: will be in terms of the number of iterations, and each iteration evaluates $\mathrm{prox}_{h,t}(\cdot)$ once (this can be cheap or expensive, depending on $h$)

# Example: ISTA

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

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

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

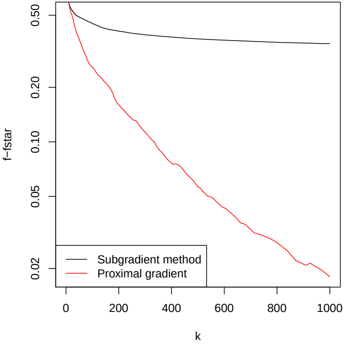
$$[S_\lambda(\beta)]_i = \begin{cases} \beta_i - \lambda & \text{if } \beta_i > \lambda \\ 0 & \text{if } -\lambda \le \beta_i \le \lambda\,, \\ \beta_i + \lambda & \text{if } \beta_i < -\lambda \end{cases} \quad i = 1, \ldots, n$$

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 curves




[1] 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 $\nabla g$ 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 $\beta/L$

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

# 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:

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

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

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

where $r = \mathrm{rank}(B)$ and $\sigma_1(X) \geq \cdots \geq \sigma_r(X) \geq 0$ are the singular values

Define $P_\Omega$, 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

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

Two ingredients needed for proximal gradient descent:

- Gradient calculation: $\nabla g(B) = -(P_\Omega(Y) - P_\Omega(B))$
- Prox function:

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

Claim: $\mathrm{prox}_t(B) = S_{\lambda t}(B)$, 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 $\Sigma_\lambda$ 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

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

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 $f = g + h$:

- $h = 0$: gradient descent
- $h = I_C$: projected gradient descent
- $g = 0$: proximal minimization algorithm

# 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

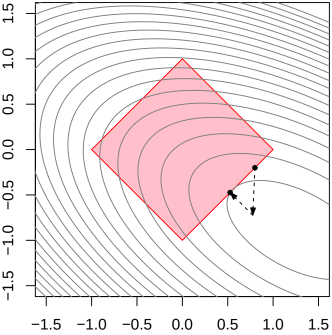
$$\begin{aligned} \mathrm{prox}_t(x) &= \underset{z}{\operatorname{argmin}} \; \frac{1}{2t} \|x - z\|_2^2 + I_C(z) \\ &= \underset{z \in C}{\operatorname{argmin}} \; \|x - z\|_2^2 \end{aligned}$$

That is, $\mathrm{prox}_t(x) = P_C(x)$, projection operator onto $C$

Therefore proximal gradient update step is:

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

That is, 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

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

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), an 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

As before, consider:

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

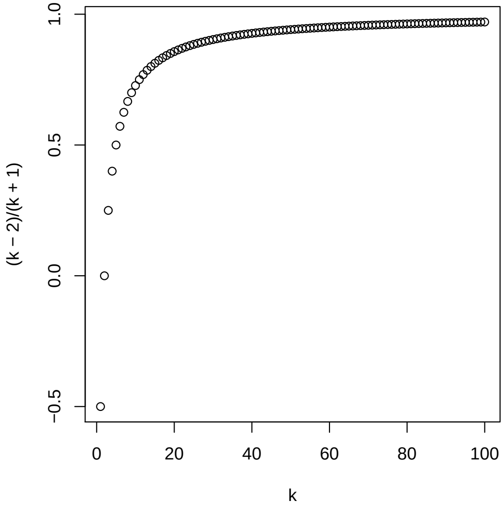
where $g$ convex, differentiable, and $h$ convex. Accelerated proximal gradient method: choose initial point $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$, repeat:

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

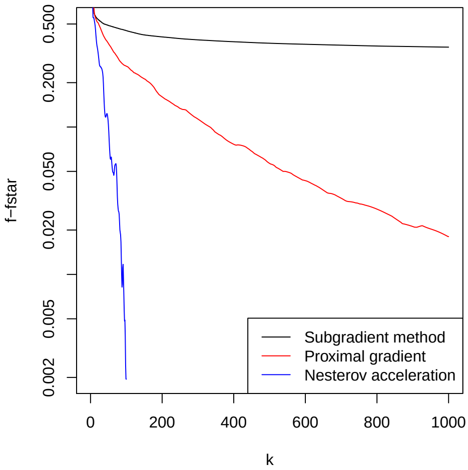
for $k = 1, 2, 3, \ldots$

- 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
- 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

# 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 $\nabla g$ 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 $t \le 1/L$ satisfies

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

and same result holds for backtracking, with $t$ replaced by $\beta/L$

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):

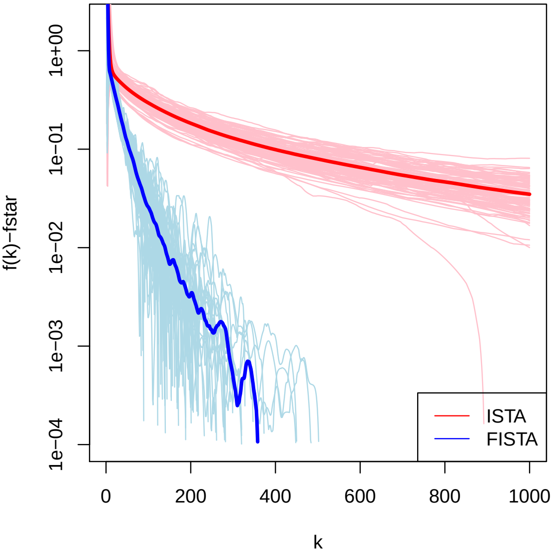
$$\beta^{(k)} = S_{\lambda t_k}\big(\beta^{(k-1)} + t_k X^T(y - X\beta^{(k-1)})\big), \quad k = 1, 2, 3, \ldots$$

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

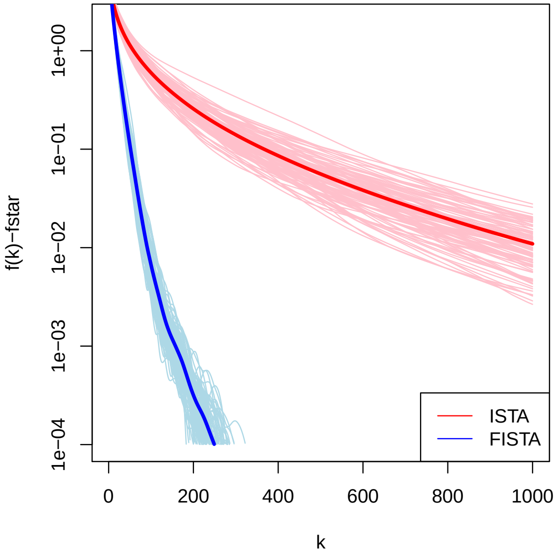
$$\begin{aligned} v &= \beta^{(k-1)} + \frac{k-2}{k+1}(\beta^{(k-1)} - \beta^{(k-2)}) \\ \beta^{(k)} &= S_{\lambda t_k}\big(v + t_k X^T(y - Xv)\big), \end{aligned}$$

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



# 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. For example, suppose want to solve lasso problem for tuning parameters values

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

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

Over a fine enough grid of $\lambda$ 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 $S_\lambda$ is the matrix soft-thresholding operator ... requires SVD

- One backtracking loop evaluates prox, 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 $t = 1$),

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

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

# 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