Generalized gradient descent

Geoff Gordon & Ryan Tibshirani
Optimization 10-725 / 36-725

Remember subgradient method

We want to solve

$$\min_{x \in \mathbb{R}^n} f(x),$$

for f convex, not necessarily differentiable

Subgradient method: choose initial $x^{(0)} \in \mathbb{R}^n$, repeat:

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

where $g^{(k-1)}$ is a subgradient of f at $x^{(k-1)}$

If f is Lipschitz on a bounded set containing its minimizer, then subgradient method has convergence rate $O(1/\sqrt{k})$

Downside: can be very slow!

Outline

Today:

- Generalized gradient descent
- Convergence analysis
- ISTA, matrix completion
- Special cases

Decomposable functions

Suppose

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

- g is convex, differentiable
- h is convex, not necessarily differentiable

If f were differentiable, gradient descent update:

$$x^+ = x - t\nabla f(x)$$

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

$$x^+ = \operatorname*{argmin}_z \underbrace{f(x) + \nabla f(x)^T(z - x) + \frac{1}{2t}\|z - x\|^2}_{\widehat{f}_t(z)}$$

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

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

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

Generalized gradient descent

Define

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

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

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

To make update step look familiar, can write it as

$$x^{(k)} = x^{(k-1)} - t_k \cdot G_{t_k}(x^{(k-1)})$$

where G_t is the generalized gradient,

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

What good did this do?

You have a right to be suspicious ... looks like we just swapped one minimization problem for another

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

- prox_t doesn't depend on g at all
- g can be very complicated as long as we can compute its gradient

Convergence analysis: will be in terms of $\#$ of iterations of the algorithm

Each iteration evaluates $\text{prox}_t(\cdot)$ once, and this can be cheap or expensive, depending on h

ISTA

Consider lasso criterion

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

Prox function is now

$$\begin{aligned} \mathrm{prox}_t(x) &= \underset{z \in \mathbb{R}^n}{\mathrm{argmin}} \ \frac{1}{2t}\|x - z\|^2 + \lambda\|z\|_1 \\ &= S_{\lambda t}(x) \end{aligned}$$

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

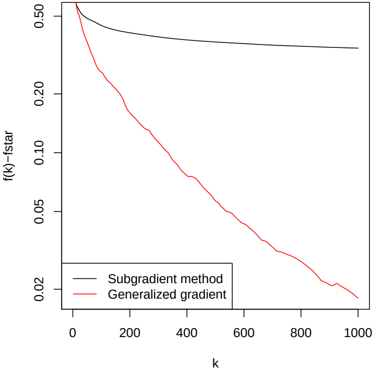
$$[S_\lambda(x)]_i = \begin{cases} x_i - \lambda & \text{if } x_i > \lambda \\ 0 & \text{if } -\lambda \leq x_i \leq \lambda \\ x_i + \lambda & \text{if } x_i < -\lambda \end{cases}$$

Recall $\nabla g(x) = -A^T(y - Ax)$. Hence generalized gradient update step is:

$$x^+ = S_{\lambda t}(x + tA^T(y - Ax))$$

Resulting algorithm called **ISTA** (Iterative Soft-Thresholding Algorithm). Very simple algorithm to compute a lasso solution

Generalized gradient (ISTA) vs subgradient descent:



Convergence analysis

We have $f(x) = g(x) + h(x)$, and assume

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

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

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

I.e., generalized gradient descent has convergence rate $O(1/k)$

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

Proof

Similar to proof for gradient descent, but with generalized gradient G_t replacing gradient ∇f. Main steps:

- ∇g Lipschitz with constant $L \Rightarrow$

$$f(y) \le g(x) + \nabla g(x)^T (y-x) + \frac{L}{2}\|y-x\|^2 + h(y) \quad \text{all } x, y$$

- Plugging in $y = x^+ = x - tG_t(x)$,

$$f(x^+) \le g(x) - t\nabla g(x)^T G_t(x) + \frac{Lt}{2}\|G_t(x)\|^2 + h(x - tG_t(x))$$

- By definition of prox,

$$\begin{aligned} x - tG_t(x) &= \operatorname*{argmin}_{z \in \mathbb{R}^n} \frac{1}{2t}\|z - (x - t\nabla g(x))\|^2 + h(z) \\ \Rightarrow \quad & \nabla g(x) - G_t(x) + v = 0, \quad v \in \partial h(x - tG_t(x)) \end{aligned}$$

- Using $G_t(x) - \nabla g(x) \in \partial h(x - tG_t(x))$, and convexity of g,

$$f(x^+) \le f(z) + G_t(x)^T(x - z) - (1 - \frac{Lt}{2})t\|G_t(x)\|^2 \quad \text{all } z$$

- Letting $t \le 1/L$ and $z = x^\star$,

$$\begin{aligned} f(x^+) &\le f(x^\star) + G_t(x)^T(x^\star - x) - \frac{t}{2}\|G_t(x)\|^2 \\ &= f(x^\star) + \frac{1}{2t}\big(\|x - x^\star\|^2 - \|x^+ - x^\star\|^2\big) \end{aligned}$$

Proof proceeds just as with gradient descent. $\square$

Backtracking line search

Same as with gradient descent, just replace ∇f with generalized gradient G_t. I.e.,

- Fix $0 < \beta < 1$
- Then at each iteration, start with $t = 1$, and while

$$f(x - tG_t(x)) > f(x) - \frac{t}{2}\|G_t(x)\|^2,$$

  update $t = \beta t$

Theorem: Generalized gradient descent with backtracking line search satisfies

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

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

Matrix completion

Given matrix A, $m \times n$, only observe entries $A_{ij}, (i,j) \in \Omega$

Want to fill in missing entries (e.g., NETFLIX), so we solve:

$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2} \sum_{(i,j) \in \Omega} (A_{ij} - X_{ij})^2 + \lambda \|X\|_*$$

Here $\|X\|_*$ is the **nuclear norm** of X,

$$\|X\|_* = \sum_{i=1}^{r} \sigma_i(X)$$

where $r = \operatorname{rank}(X)$ and $\sigma_1(X), \ldots \sigma_r(X)$ are its singular values

Define P_Ω, projection operator onto observed set:

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

Criterion is

$$f(X) = \underbrace{\frac{1}{2}\|P_\Omega(A) - P_\Omega(X)\|_F^2}_{g(X)} + \underbrace{\lambda\|X\|_*}_{h(X)}$$

Two things for generalized gradient descent:

- Gradient: $\nabla g(X) = -(P_\Omega(A) - P_\Omega(X))$
- Prox function:

$$\mathrm{prox}_t(X) = \underset{Z \in \mathbb{R}^{m \times n}}{\mathrm{argmin}} \; \frac{1}{2t}\|X - Z\|_F^2 + \lambda\|Z\|_*$$

Claim: $\mathrm{prox}_t(X) = S_{\lambda t}(X)$, where the **matrix soft-thresholding** operator $S_\lambda(X)$ is defined by

$$S_\lambda(X) = U\Sigma_\lambda V^T$$

where $X = U\Sigma V^T$ is a singular value decomposition, and Σ_λ is diagonal with

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

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

$$0 \in Z - X + \lambda t \cdot \partial \|Z\|_*$$

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

$$\partial \|Z\|_* = \{UV^T + W : W \in \mathbb{R}^{m\times n}, \|W\| \le 1, U^T W = 0, WV = 0\}$$

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

Hence generalized gradient update step is:

$$X^+ = S_{\lambda t}(X + t(P_\Omega(A) - P_\Omega(X)))$$

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

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

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

This is the **soft-impute** algorithm[1], simple and effective method for matrix completion

[1]Mazumder et al. (2011), *Spectral regularization algorithms for learning large incomplete matrices*

Why "generalized"?

Special cases of generalized gradient descent, on $f = g + h$:

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

Therefore these algorithms all have $O(1/k)$ convergence rate

Projected gradient descent

Given closed, convex set $C \in \mathbb{R}^n$,

$$\min_{x \in C} g(x) \quad \Leftrightarrow \quad \min_{x} g(x) + I_C(x)$$

where $I_C(x) = \begin{cases} 0 & x \in C \\ \infty & x \notin C \end{cases}$ is the indicator function of C

Hence

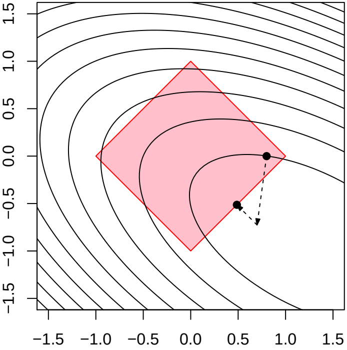
$$\begin{aligned} \mathrm{prox}_t(x) &= \operatorname*{argmin}_z \frac{1}{2t}\|x - z\|^2 + I_C(z) \\ &= \operatorname*{argmin}_{z \in C} \|x - z\|^2 \end{aligned}$$

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

Therefore generalized gradient update step is:

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

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



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

- Affine images $C = \{Ax + b : x \in \mathbb{R}^n\}$
- Solution set of linear system $C = \{x \in \mathbb{R}^n : Ax = b\}$
- Nonnegative orthant $C = \{x \in \mathbb{R}^n : x \geq 0\} = \mathbb{R}^n_+$
- Norm balls $C = \{x \in \mathbb{R}^n : \|x\|_p \leq 1\}$, 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., it is generally hard to project onto solution set of arbitrary linear inequalities, i.e, arbitrary polyhedron $C = \{x \in \mathbb{R}^n : Ax \leq b\}$

Proximal minimization algorithm

Consider for h convex (not necessarily differentiable),

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

Generalized gradient update step is just a prox update:

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

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

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

can be done exactly

Generally speaking, all bets are off if we just treat this as another minimization problem, and obtain an approximate solution. And practical convergence can be very slow if we use an approximation to the prox

But there are exceptions (both in theory and in practice), e.g., partial proximation minimization[2]

[2] Bertsekas and Tseng (1994), *Partial proximal minimization algorithms for convex programming*

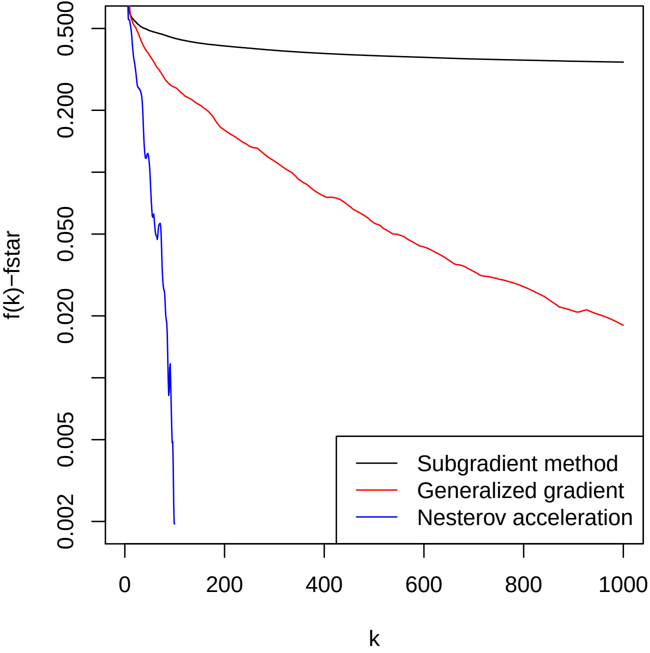
Almost cutting edge

We're almost at the cutting edge for first order methods, but not quite ... still require too many iterations

Acceleration: use more than just $x^{(k-1)}$ to compute $x^{(k)}$ (e.g., use $x^{(k-2)}$), sometimes called momentum terms or memory terms

There are many different flavors of acceleration (at least three, mostly due to Nesterov)

Accelerated generalized gradient descent achieves optimal rate $O(1/k^2)$ among first order methods for minimizing $f = g + h$!



0.500
0.200
0.050
0.020
0.005
0.002
f(k)−fstar
0
200
400
600
800
1000
k
Subgradient method
Generalized gradient
Nesterov acceleration

References

- E. Candes, Lecture Notes for Math 301, Stanford University, Winter 2010-2011
- L. Vandenberghe, Lecture Notes for EE 236C, UCLA, Spring 2011-2012