Sparse Optimization
Lecture: Proximal Operator/Algorithm and Lagrange Dual

Instructor: Wotao Yin

July 2013

online discussions on piazza.com

Those who complete this lecture will know

- learn the proximal operator and its basic properties
- the proximal algorithm
- the proximal algorithm applied to the Lagrange dual

Gradient descent / forward Euler

- assume function f is convex, differentiable
- consider

$$\min f(\mathbf{x})$$

- gradient descent iteration (with step size c):

$$\mathbf{x}^{k+1} = \mathbf{x}^k - c\,\nabla f(\mathbf{x}^k)$$

- $\mathbf{x}^{k+1}$ *minimizes* the following local quadratic approximation of f:

$$f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{1}{2c}\|\mathbf{x} - \mathbf{x}^k\|_2^2$$

- compare with forward Euler iteration, a.k.a. the explicit update:

$$\mathbf{x}(t+1) = \mathbf{x}(t) - \Delta t \cdot \nabla f(\mathbf{x}(t))$$

Backward Euler / implicit gradient descent

- backward Euler iteration, also known as the implicit update:

$$\mathbf{x}(t+1) \xleftarrow{\text{solve}} \mathbf{x}(t+1) = \mathbf{x}(t) - \Delta t \cdot \nabla f\left(\mathbf{x}(t+1)\right).$$

- equivalent to:
  1. $\mathbf{u}(t+1) \xleftarrow{\text{solve}} \mathbf{u} = \nabla f\left(\mathbf{x}(t) - \Delta t \cdot \mathbf{u}\right),$
  2. $\mathbf{x}(t+1) = \mathbf{x}(t) - \Delta t \cdot \mathbf{u}(t+1).$

- we can view it as the "implicit gradient" descent:

$$(x^{k+1}, u^{k+1}) \xleftarrow{\text{solve}} x = x^k - cu, \;\; u = \nabla f(x).$$

- c is the "step size", very different from a standard step size.
- explicit (implicit) update uses the gradient at the start (end) point

Implicit gradient step = proximal operation

- proximal update:

$$\mathbf{prox}_{cf}(\mathbf{z}) := \arg\min_{x} f(\mathbf{x}) + \frac{1}{2c}\|\mathbf{x}-\mathbf{z}\|^2.$$

- optimality condition:

$$0 = c\,\nabla f(\mathbf{x}^*) + (\mathbf{x}^* - \mathbf{z}).$$

- given input $\mathbf{z}$,
  - $\mathbf{prox}_{cf}(\mathbf{z})$ returns solution x^*
  - $\nabla f(\mathbf{prox}_{cf}(z))$ returns u^*

$$(\mathbf{x}^*, \mathbf{u}^*) \xleftarrow{\text{solve}} \mathbf{x} = \mathbf{z} - c\mathbf{u}, \;\; \mathbf{u} = \nabla f(\mathbf{x}).$$

Proposition

Proximal operator is equivalent to an implicit gradient (or backward Euler) step.

Proximal operator handles sub-differentiable f

- assume that f is closed, proper, *sub-differentiable* convex function
- $\partial f(\mathbf{x})$ is denoted as the subdifferential of f at x.
  Recall $\mathbf{u} \in \partial f(\mathbf{x})$ if

$$f(\mathbf{x}') \geq f(\mathbf{x}) + \langle \mathbf{u}, \mathbf{x}' - \mathbf{x} \rangle, \ \forall \mathbf{x}' \in \mathbb{R}^n.$$

- $\partial f(\mathbf{x})$ is point-to-set, neither direction is unique
- $\mathbf{prox}$ is well-defined for sub-differentiable f; it is point-to-point,
  $\mathbf{prox}$ maps any input to a unique point

Proximal operator

$$\mathbf{prox}_{cf}(\mathbf{z}) := \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{1}{2c}\|\mathbf{x}-\mathbf{z}\|^2.$$

- since objective is strongly convex, solution $\mathbf{prox}_{cf}(\mathbf{z})$ is unique
- since f is proper, $\mathrm{dom}\ \mathbf{prox}_{cf} = \mathbb{R}^n$
- the followings are equivalent

$$\mathbf{prox}_{cf}\mathbf{z} = \mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{1}{2c}\|\mathbf{x}-\mathbf{z}\|^2,$$

$$\mathbf{x}^* \xleftarrow{\text{solve}} 0 \in c\,\partial f(\mathbf{x}) + (\mathbf{x}-\mathbf{z}),$$

$$(\mathbf{x}^*, \mathbf{u}^*) \xleftarrow{\text{solve}} \mathbf{x} = \mathbf{z} - c\mathbf{u},\ \ \mathbf{u} \in \partial f(\mathbf{x}).$$

- point $\mathbf{x}^*$ minimizes f if and only if $\mathbf{x}^* = \mathbf{prox}_f(\mathbf{x}^*)$.

Examples

- $f(\mathbf{x}) = \iota_{\mathbf{x} \in \mathcal{C}}$
- $f(\mathbf{x}) = \frac{\lambda}{2} \|\mathbf{x}\|_2^2$
- $f(\mathbf{x}) = \|\mathbf{x}\|_1$
- $f(\mathbf{x}) = \sum_i \|\mathbf{x}_{\mathcal{G}_i}\|_2$
- $f(\mathbf{X}) = \|\mathbf{X}\|_*$

Examples

given $\mathbf{prox}_f$ for function f, it is easy to derive $\mathbf{prox}_g$ for

- $g(\mathbf{x}) = \alpha f(\mathbf{x}) + \beta$
- $g(\mathbf{x}) = f(\alpha \mathbf{x} + \mathbf{b})$
- $g(\mathbf{x}) = f(\mathbf{x}) + \mathbf{a}^T \mathbf{x} + \beta$
- $g(\mathbf{x}) = f(\mathbf{x}) + (\rho/2)\|\mathbf{x} - \mathbf{a}\|^2$

Resolvent of ∂f

- ∂f is a *point-to-set* mapping, so is $I + c\,\partial f$
- in general, $(I + c\,\partial f)^{-1}$ is a *point-to-set* mapping
- however, we claim

$$\mathbf{prox}_{cf} = (I + c\,\partial f)^{-1}$$

- since $\mathbf{prox}_{cf}(\mathbf{z})$ is always unique, $(I + c\,\partial f)^{-1}$ is a *point-to-point* mapping

- $(I + c\,\partial f)^{-1}$ is known as the *resolvent* of ∂f with parameter c.
- by the way, ∇f is the gradient operator, and $(I - c\,\nabla f)$ is the *gradient-descent* operator.

Moreau envelope

- **idea**: to smooth a closed, proper, *nonsmooth* convex function f
- definition:
$$M_{cf}(\mathbf{x}) = \inf_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2c}\|\mathbf{y} - \mathbf{x}\|^2.$$
- $\operatorname{dom} M_{cf} = \mathbb{R}^n$ even if f is not
- $M_{cf} \in C^1$ even if f is not; in fact,
$$M_{cf} = \left((cf)^* + (1/2)\|\cdot\|^2\right)^*$$
the dual of strongly convex function is differentiable (with Lipschitz gradient)
- relation with $\mathbf{prox}_{cf}$
  - $\nabla M_{cf}(\mathbf{x}) = (1/c)(\mathbf{x} - \mathbf{prox}_{cf}(\mathbf{x}))$
  - $\mathbf{prox}_{cf}(\mathbf{x}) = \mathbf{x} - c\nabla M_{cf}(\mathbf{x})$, explicit gradient step of M_{cf}
  - $\mathbf{prox}_f(\mathbf{x}) = \nabla M_{f^*}(\mathbf{x})$
- example: the Huber function is M_f where $f(\mathbf{x}) = \|\mathbf{x}\|_1$
- c is not a usual step size. As $c \to \infty$, $(\mathbf{prox}_{cf}(\mathbf{x}) - \mathbf{x}) \to (\mathbf{x}^* - \mathbf{x})$.

Proximal algorithm

Assume that f has a minimizer, then iterate

$$\mathbf{x}^{k+1} = \mathbf{prox}_{c^k f}(\mathbf{x}^k)$$

$\mathbf{prox}$ is *firmly nonexpansive*

$$\|\mathbf{prox}_f(\mathbf{x}) - \mathbf{prox}_f(\mathbf{y})\|^2 \le \langle \mathbf{prox}_f(\mathbf{x}) - \mathbf{prox}_f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle,\ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

It converges to the minimizer as long as $c^k > 0$ and

$$\sum_{k=1}^{\infty} c^k = \infty.$$

For example, one can fix $c^k \equiv c$

Step-sized iteration: fix $c > 0$ and pick $\alpha_k \in (0, 2)$ uniformly away from 0 and 2:

$$\mathbf{x}^{k+1} = \alpha_k \mathbf{prox}_{cf}(\mathbf{x}^k) + (1 - \alpha^k)\mathbf{x}^k.$$

The convergence takes a *finite* number of iterations if f is polyhedral (i.e. piece-wise linear)

Proximal algorithm

Diminishing regularization

$$\mathbf{x}^{k+1} = \arg\min f(\mathbf{x}) + \frac{1}{2c}\|\mathbf{x} - \mathbf{x}^k\|_2^2$$

As $\mathbf{x}^k \to \mathbf{x}^*$, $\|\partial f(\mathbf{x}^k)\| \to 0$ and thus $f(\mathbf{x})$ becomes "weaker." Hence, $\mathbf{x}^{k+1} - \mathbf{x}^k$ tends to be smaller.

Many algorithms use $\mathbf{prox}_{c^k f}(x^k)$, either entirely or as a part (but most of them were motivated through other means)

Although $\mathbf{prox}_{c^k f}$ can sometimes be difficult to compute, it simplifies computation

- for some sub-differentiable functions
- for those rising in duality (our next focus)

Lagrange duality

Convex problem

$$\boxed{\min f(\mathbf{x}) \quad \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}.}$$

Relax the constraints and price their violation (pay a price if violated one way; get paid if violated the other way; payment is linear to the violation)

$$\mathcal{L}(\mathbf{x}; \mathbf{y}) := f(\mathbf{x}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$$

For *later use*, define the *augmented Lagrangian*

$$\mathcal{L}_A(\mathbf{x}; \mathbf{y}, \mathbf{c}) := f(\mathbf{x}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{c}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$$

Minimize $\mathcal{L}$ for fixed price $\mathbf{y}$: $d(\mathbf{y}) := -\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y})$. Always, $d(\mathbf{y})$ is convex

The Lagrange dual problem

$$\boxed{\min_{\mathbf{y}} d(\mathbf{y})}$$

Given dual solution $\mathbf{y}^*$, recover $\mathbf{x}^* = \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^*)$ (under which conditions?)

Question: how to compute the explicit/implicit gradients of $d(\mathbf{y})$?

Dual explicit gradient (ascent) algorithm

Assume $d(\mathbf{y})$ is differentiable (true if $f(\mathbf{x})$ is strictly convex. Is this if-and-only-if?)

Gradient descent iteration (if the maximizing dual is used, it is called *gradient ascent*):

$$\mathbf{y}^{k+1} = \mathbf{y}^{k} - c\,\nabla f(\mathbf{y}^{k}).$$

It turns out to be *relatively easy* to compute ∇d, via an unstrained subproblem:

$$\nabla d(\mathbf{y}) = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}, \quad \text{where } \bar{\mathbf{x}} = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}).$$

Dual gradient iteration

1. $\mathbf{x}^k \xleftarrow{\text{solve}} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^k)$;
2. $\mathbf{y}^{k+1} = \mathbf{y}^k - c(\mathbf{b} - \mathbf{A}\mathbf{x}^k)$.

Sub-gradient of $d(\mathbf{y})$

Assume $d(\mathbf{y})$ is sub-differentiable (which condition on primal can guarantee this?)

Lemma

Given dual point $\mathbf{y}$ and $\bar{\mathbf{x}} = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y})$, we have $\mathbf{b} - \mathbf{A}\bar{\mathbf{x}} \in \partial d(\mathbf{y})$.

Proof.

Recall

- $\mathbf{u} \in \partial d(\mathbf{y})$ if $d(\mathbf{y}') \geq d(\mathbf{y}) + \langle \mathbf{u}, \mathbf{y}' - \mathbf{y} \rangle$ for all $\mathbf{y}'$;
- $d(\mathbf{y}) := -\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y})$.

From (ii) and definition of $\bar{\mathbf{x}}$,

$$
\begin{aligned}
d(\mathbf{y}) + \langle \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}, \mathbf{y}' - \mathbf{y} \rangle &= -\mathcal{L}(\bar{\mathbf{x}}; \mathbf{y}) + (\mathbf{b} - \mathbf{A}\bar{\mathbf{x}})^T (\mathbf{y} - \mathbf{y}') \\
&= -[f(\bar{\mathbf{x}} + \mathbf{y}^T (\mathbf{A}\bar{\mathbf{x}} - \mathbf{b})] + (\mathbf{b} - \mathbf{A}\bar{\mathbf{x}})^T (\mathbf{y} - \mathbf{y}') \\
&= -[f(\bar{\mathbf{x}}) + (\mathbf{y}')^T (\mathbf{A}\bar{\mathbf{x}} - \mathbf{b})] \\
&= -\mathcal{L}(\bar{\mathbf{x}}; \mathbf{y}') \leq d(\mathbf{y}').
\end{aligned}
$$

From (i), $\mathbf{b} - \mathbf{A}\bar{\mathbf{x}} \in \partial d(\mathbf{y})$. $\square$

Dual explicit (sub)gradient iteration

The iteration:

1. $\mathbf{x}^k \xleftarrow{\text{solve}} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^k)$;
2. $\mathbf{y}^{k+1} = \mathbf{y}^k - c_k(\mathbf{b} - \mathbf{A}\mathbf{x}^k)$;

Notes:

- $(\mathbf{b} - \mathbf{A}\mathbf{x}^k) \in \partial d(\mathbf{y}^k)$ as shown in the last slide
- it does *not* require $d(\mathbf{y})$ to be differentiable
- convergence might require a careful choice of c_k (e.g., a diminishing sequence) if $d(\mathbf{y})$ is only sub-differentiable (or lacking Lipschitz continuous gradient)

Dual implicit gradient

Goal: to descend using the (sub)gradient of d at the *next point* y^{k+1}:

Following from the Lemma, we have

$$\mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \in \partial d(\mathbf{y}^{k+1}), \text{ where } \mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^{k+1})$$

Since the implicit step is $\mathbf{y}^{k+1} = \mathbf{y}^k - c(\mathbf{b} - A\mathbf{x}^{k+1})$, we can derive

$$\begin{aligned}
\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^{k+1}) &\iff \\
0 \in \partial_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{k+1}; \mathbf{y}^{k+1}) &= \partial f(\mathbf{x}^{k+1}) + \mathbf{A}^T \mathbf{y}^{k+1} \\
&= \partial f(\mathbf{x}^{k+1}) + \mathbf{A}^T(\mathbf{y}^k - c(\mathbf{b} - \mathbf{A}\mathbf{x}^{k+1})).
\end{aligned}$$

Therefore, while $\mathbf{x}^{k+1}$ is a solution to $\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^{k+1})$; it is also a solution to

$$\min_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}; \mathbf{y}^k, c) = f(\mathbf{x}) + (\mathbf{y}^k)^T(\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{c}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2,$$

which is *independent* of $\mathbf{y}^{k+1}$.

Dual implicit gradient

Proposition

Assuming $\mathbf{y}' = \mathbf{y} - c(\mathbf{b} - \mathbf{A}\mathbf{x}')$, *the followings are equivalent*

1. $\mathbf{x}' \stackrel{\text{solve}}{\longleftarrow} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}')$,
2. $\mathbf{x}' \stackrel{\text{solve}}{\longleftarrow} \min_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}; \mathbf{y}, c)$.

Dual implicit gradient iteration

The iteration

$$\mathbf{y}^{k+1} = \mathbf{prox}_{cd}(\mathbf{y}^k)$$

is commonly known as **the augmented Lagrangian method** or **the method of multipliers**.

Implementation:

1. $\mathbf{x}^{k+1} \overset{\text{solve}}{\longleftarrow} \min_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}; \mathbf{y}^k, c)$;
2. $\mathbf{y}^{k+1} = \mathbf{y}^k - c(\mathbf{b} - \mathbf{A}\mathbf{x}^{k+1})$.

Proposition

The followings are equivalent

1. *the augmented Lagrangian iteration;*
2. *the implicit gradient iteration of $d(\mathbf{y})$;*
3. *the proximal iteration $\mathbf{y}^{k+1} = \mathbf{prox}_{cd}(\mathbf{y}^k)$.*

Dual explicit/implicit (sub)gradient computation

Definitions:

- $\mathcal{L}(x; y) = f(x) + y^T(Ax - b)$
- $\mathcal{L}_A(x; y, c) = \mathcal{L}(x; y) + \frac{c}{2}\|Ax - b\|^2$

Objective:

$$d(y) = -\min_x \mathcal{L}(x; y).$$

Explicit (sub)gradient iteration: $\mathbf{y}^{k+1} = \mathbf{y}^k - c\nabla d(\mathbf{y}^k)$ or use a subgradient $\partial d(\mathbf{y}^k)$

1. $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^k)$;
2. $\mathbf{y}^{k+1} = \mathbf{y}^k - c(\mathbf{b} - \mathbf{A}\mathbf{x}^{k+1})$.

Implicit (sub)gradient step: $\mathbf{y}^{k+1} = \mathbf{prox}_{cd}\mathbf{y}^k$

1. $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}; \mathbf{y}^k, c)$;
2. $\mathbf{y}^{k+1} = \mathbf{y}^k - c(\mathbf{b} - \mathbf{A}\mathbf{x}^{k+1})$.

The implicit iteration is more stable; "step size" c does not need to diminish.