ORIE 4741: Learning with Big Messy Data

Proximal Gradient Method

Professor Udell

Operations Research and Information Engineering
Cornell

October 16, 2021

Announcements

- Homework 5 due next Thursday 11/14
- Pick up midterm exams from Prof. Udell's office hours
- Bug fix!
  - update package repository:

    ```
    using Pkg
    Pkg.update()
    ```

  - download new version of `proxgrad.jl` from demos repository

Demo: proximal gradient

`https://github.com/ORIE4741/demos/blob/master/proxgrad-starter-code.ipynb`

Regularized empirical risk minimization

choose model by solving

$$\text{minimize} \quad \sum_{i=1}^{n} \ell(x_i, y_i; w) + r(w)$$

with variable $w \in \mathbf{R}^d$

- parameter vector $w \in \mathbf{R}^d$
- loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathbf{R}^d \to \mathbf{R}$
- regularizer $r : \mathbf{R}^d \to \mathbf{R}$

Regularized empirical risk minimization

choose model by solving

$$\text{minimize} \quad \sum_{i=1}^{n} \ell(x_i, y_i; w) + r(w)$$

with variable $w \in \mathbf{R}^d$

- parameter vector $w \in \mathbf{R}^d$
- loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathbf{R}^d \to \mathbf{R}$
- regularizer $r : \mathbf{R}^d \to \mathbf{R}$

why?

- want to minimize the **risk** $\mathbb{E}_{(x,y)\sim P}\ell(x, y; w)$
- approximate it by the **empirical risk** $\sum_{i=1}^{n} \ell(x, y; w)$
- add regularizer to help model generalize

Solving regularized risk minimization

how should we fit these models?

- with a different software package for each model?
- with a different algorithm for each model?
- with a general purpose optimization solver?

desiderata

- fast
- flexible

What's wrong with gradient descent?

Q: Why can't we use gradient descent to solve all our problems?

What's wrong with gradient descent?

Q: Why can't we use gradient descent to solve all our problems?
A: Because some regularizers and loss functions aren't differentiable!

Subgradient

Definition

The vector $g \in \mathbf{R}^d$ is a **subgradient** of $f : \mathbf{R}^d \to \mathbf{R}$ at x if

$$f(y) \geq f(x) + g^\top (y - x), \quad \forall y \in \mathbf{R}^d.$$

The **subdifferential** of $f : \mathbf{R}^d \to \mathbf{R}$ at x is the set of all subgradients of f at x:

$$\partial f(x) = \{g : f(y) \geq f(x) + g^\top (y - x) \ \forall y \in \mathbf{R}^d\}$$

Subgradient

Definition

The vector $g \in \mathbf{R}^d$ is a **subgradient** of $f : \mathbf{R}^d \to \mathbf{R}$ at x if

$$f(y) \geq f(x) + g^\top (y - x), \quad \forall y \in \mathbf{R}^d.$$

The **subdifferential** of $f : \mathbf{R}^d \to \mathbf{R}$ at x is the set of all subgradients of f at x:

$$\partial f(x) = \{g : f(y) \geq f(x) + g^\top (y - x)\ \forall y \in \mathbf{R}^d\}$$

- one subgradient for each supporting hyperplane of f at x

Subgradient

Definition

The vector $g \in \mathbf{R}^d$ is a **subgradient** of $f : \mathbf{R}^d \to \mathbf{R}$ at x if

$$f(y) \geq f(x) + g^\top (y - x), \quad \forall y \in \mathbf{R}^d.$$

The **subdifferential** of $f : \mathbf{R}^d \to \mathbf{R}$ at x is the set of all subgradients of f at x:

$$\partial f(x) = \{g : f(y) \geq f(x) + g^\top (y - x) \ \forall y \in \mathbf{R}^d\}$$

- one subgradient for each supporting hyperplane of f at x
- the subdifferential ∂f maps points to sets

Subgradient

for $f : \mathbf{R} \to \mathbf{R}$ and convex, here's a simpler equivalent condition:

- if f is differentiable at x, $\partial f(x) = \{\nabla f(x)\}$
- if f is not differentiable at x, it will still be differentiable just to the left and the right of x[1], so
  - let $g^+ = \lim_{\epsilon \to 0} \nabla f(x + \epsilon)$
  - let $g^- = \lim_{\epsilon \to 0} \nabla f(x - \epsilon)$
  - $\partial f(x)$ is any convex combination (*i.e.*, any weighted average) of those gradients:

$$\partial f(x) = \{\alpha g^+ + (1 - \alpha) g^- : \alpha \in [0, 1]\}$$

[1](A convex function is differentiable almost everywhere.)

Subgradient: examples

compute subgradient wrt prediction vector $z \in \mathbf{R}$:

- quadratic loss: $\ell(y, z) = (y - z)^2$
- ℓ_1 loss: $\ell(y, z) = |y - z|$
- hinge loss: $\ell(y, z) = (1 - yz)_+$
- logistic loss: $\ell(y, z) = \log(1 + \exp(-yz))$

Important properties of subdifferential

- **Linearity.**

$$\partial_w \sum_{(x,y)\in\mathcal{D}} \ell(y, w^\top x) = \sum_{(x,y)\in\mathcal{D}} \partial_w \ell(y, w^\top x)$$

Important properties of subdifferential

- **Linearity.**

$$\partial_w \sum_{(x,y)\in\mathcal{D}} \ell(y, w^\top x) = \sum_{(x,y)\in\mathcal{D}} \partial_w \ell(y, w^\top x)$$

- **Chain rule.** If $f = h \circ g$, $h : \mathbf{R} \to \mathbf{R}$, and $g : \mathbf{R}^d \to \mathbf{R}$ is differentiable,

$$\partial f(x) = \partial h(g(x)) \nabla g(x).$$

Important properties of subdifferential

- **Linearity.**

$$\partial_w \sum_{(x,y)\in\mathcal{D}} \ell(y, w^\top x) = \sum_{(x,y)\in\mathcal{D}} \partial_w \ell(y, w^\top x)$$

- **Chain rule.** If $f = h \circ g$, $h : \mathbf{R} \to \mathbf{R}$, and $g : \mathbf{R}^d \to \mathbf{R}$ is differentiable,

$$\partial f(x) = \partial h(g(x)) \nabla g(x).$$

Example. For $z = w^\top x$,

$$\partial_w \ell(y, w^\top x) = \partial_z \ell(y, z) \nabla_w (w^\top x) = x \partial_z \ell(y, z)$$

Subgradient method

$$\text{minimize} \quad \ell(w)$$

Algorithm Subgradient method

Given: function $\ell : \mathbf{R}^d \to \mathbf{R}$, stepsize sequence $\{\alpha^t\}_{t=1}^{\infty}$, maxiters
Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)
For: $t = 1, \ldots,$ maxiters

- compute subgradient $g \in \partial \ell(w)$
- update w:

$$w \leftarrow w - \alpha^t g$$

Stochastic subgradient method

stochastic subgradient obeys

$$\mathbb{E}\tilde{\partial}\ell(w) \in \partial\ell(w)$$

examples: for $\ell(w) = \sum_{i=1}^{n} \ell(y_i, w^\top x_i)$,

Stochastic subgradient method

stochastic subgradient obeys

$$\mathbb{E}\tilde{\partial}\ell(w) \in \partial\ell(w)$$

examples: for $\ell(w) = \sum_{i=1}^{n} \ell(y_i, w^\top x_i)$,

- **single stochastic gradient.** pick a random example i. set $z_i = w^\top x_i$ and compute

$$\tilde{\partial}\ell(w) = n x_i \partial_z \ell(y_i, z_i)$$

Stochastic subgradient method

stochastic subgradient obeys

$$\mathbb{E}\tilde{\partial}\ell(w) \in \partial\ell(w)$$

examples: for $\ell(w) = \sum_{i=1}^n \ell(y_i, w^\top x_i)$,

- **single stochastic gradient.** pick a random example i. set $z_i = w^\top x_i$ and compute

$$\tilde{\partial}\ell(w) = n x_i \partial_z \ell(y_i, z_i)$$

- **minibatch stochastic gradient.** pick a random set of examples S. set $z_i = w^\top x_i$ for $i \in S$ and compute

$$\begin{aligned}\tilde{\partial}\ell(w) &= \frac{n}{|S|}\partial\left(\sum_{i \in S} \ell(y_i, w^\top x_i)\right) \\ &= \frac{n}{|S|}\sum_{i \in S} x_i \partial_z \ell(y_i, z_i)\end{aligned}$$

Convergence for stochastic subgradient method

suppose ℓ is convex, subdifferentiable, Lipschitz continuous.
convergence results:

- stochastic (sub)gradient, fixed step size $\alpha^t = \alpha$:
  - iterates converge quickly, then wander within a small ball
- stochastic (sub)gradient, decreasing step size $\alpha^t = 1/t$:
  - iterates converge slowly to solution

proofs: [Bertsekas, 2010] `https://arxiv.org/pdf/1507.01030v1.pdf`

What's wrong with the subgradient method?

Q: Why can't we use the **sub**gradient method to solve all our problems?

What's wrong with the subgradient method?

Q: Why can't we use the **sub**gradient method to solve all our problems?

A:

1) because some of our regularizers don't have subgradients everywhere (*e.g.*, $\mathbf{1}_+$).

2) proximal gradient is way faster.

Proximal operator

define the **proximal operator** of the function $r : \mathbf{R}^d \to \mathbf{R}$

$$\mathbf{prox}_r(z) = \operatorname*{argmin}_w \left(r(w) + \frac{1}{2}\|w - z\|_2^2\right)$$

Proximal operator

define the **proximal operator** of the function $r : \mathbf{R}^d \to \mathbf{R}$

$$\mathbf{prox}_r(z) = \operatorname*{argmin}_w \left(r(w) + \frac{1}{2}\|w - z\|_2^2\right)$$

- $\mathbf{prox}_r : \mathbf{R}^d \to \mathbf{R}^d$

Proximal operator

define the **proximal operator** of the function $r : \mathbf{R}^d \to \mathbf{R}$

$$\mathbf{prox}_r(z) = \operatorname*{argmin}_w \left(r(w) + \frac{1}{2}\|w - z\|_2^2\right)$$

- $\mathbf{prox}_r : \mathbf{R}^d \to \mathbf{R}^d$
- **generalized projection:** if $\mathbf{1}_C$ is the indicator of set C,

$$\mathbf{prox}_{\mathbf{1}_C}(w) = \Pi_C(w)$$

Proximal operator

define the **proximal operator** of the function $r : \mathbf{R}^d \to \mathbf{R}$

$$\mathbf{prox}_r(z) = \operatorname*{argmin}_w \left(r(w) + \frac{1}{2}\|w - z\|_2^2\right)$$

- $\mathbf{prox}_r : \mathbf{R}^d \to \mathbf{R}^d$
- **generalized projection:** if $\mathbf{1}_C$ is the indicator of set C,

$$\mathbf{prox}_{\mathbf{1}_C}(w) = \Pi_C(w)$$

- **implicit gradient step:** if $w = \mathbf{prox}_r(z)$ and r is smooth,

$$\begin{aligned} \nabla r(w) + w - z &= 0 \\ w &= z - \nabla r(w) \end{aligned}$$

Proximal operator

define the **proximal operator** of the function $r : \mathbf{R}^d \to \mathbf{R}$

$$\mathbf{prox}_r(z) = \operatorname*{argmin}_w (r(w) + \frac{1}{2}\|w - z\|_2^2)$$

- $\mathbf{prox}_r : \mathbf{R}^d \to \mathbf{R}^d$
- **generalized projection:** if $\mathbf{1}_C$ is the indicator of set C,

$$\mathbf{prox}_{\mathbf{1}_C}(w) = \Pi_C(w)$$

- **implicit gradient step:** if $w = \mathbf{prox}_r(z)$ and r is smooth,

$$\begin{aligned} \nabla r(w) + w - z &= 0 \\ w &= z - \nabla r(w) \end{aligned}$$

- **simple to evaluate:** closed form solutions for many functions

more info: [Parikh Boyd 2013]

Maps from functions to functions

no consistent notation for map from functions to functions.

for a function $f : \mathbf{R}^d \to \mathbf{R}$,

- **prox** maps f to a new function $\mathbf{prox}_f : \mathbf{R}^d \to \mathbf{R}^d$
  - $\mathbf{prox}_f(x)$ evaluates this function at the point x
- ∇ maps f to a new function $\nabla f : \mathbf{R}^d \to \mathbf{R}^d$
  - $\nabla f(x)$ evaluates this function at the point x
- $\frac{\partial}{\partial x}$ maps f to a new function $\frac{\partial f}{\partial x} : \mathbf{R}^d \to \mathbf{R}^d$
  - $\frac{\partial f}{\partial x}(x)|_{x=\bar{x}}$ evaluates this function at the point $\bar{x}$
  - this one has the most confusing notation of all...

Let's evaluate some proximal operators!

define the **proximal operator** of the function $r : \mathbf{R}^d \to \mathbf{R}$

$$\mathbf{prox}_r(z) = \underset{w}{\operatorname{argmin}}(r(w) + \frac{1}{2}\|w - z\|_2^2)$$

- $r(w) = 0$ (identity)
- $r(w) = \sum_{i=1}^{d} r_i(w_i)$ (separable)
- $r(w) = \|w\|_2^2$ (shrinkage)
- $r(w) = \|w\|_1$ (soft-thresholding)
- $r(w) = \mathbf{1}(w \geq 0)$ (projection)
- $r(w) = \sum_{i=1}^{d-1}(w_{i+1} - w_i)^2$ (smoothing)

Proximal (sub)gradient method

want to solve

$$\text{minimize} \quad \ell(w) + r(w)$$

- $\ell : \mathbf{R}^d \to \mathbf{R}$ subdifferentiable
- $r : \mathbf{R}^d \to \mathbf{R}$ with a fast prox operator

---

Algorithm Proximal (sub)gradient method

Given: loss $\ell : \mathbf{R}^d \to \mathbf{R}$, regularizer $r : \mathbf{R}^d \to \mathbf{R}$, stepsizes $\{\alpha^t\}_{t=1}^{\infty}$, maxiters
Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)
For: $t = 1, \ldots,$ maxiters

- compute subgradient $g \in \partial\ell(w)$ ($\mathcal{O}(nd)$ flops)
- update w: ($\mathcal{O}(d)$ flops)

$$w \leftarrow \mathbf{prox}_{\alpha^t r}(w - \alpha^t g)$$

Example: NNLS

$$\text{minimize} \quad \frac{1}{2}\|y - Xw\|^2 + \mathbf{1}(w \geq 0)$$

recall

- $\nabla\left(\frac{1}{2}\|y - Xw\|^2\right) = -X^T(y - Xw)$
- $\mathbf{prox}_{\mathbf{1}(\cdot \geq 0)}(w) = \max(0, w)$

Algorithm Proximal gradient method for NNLS

Given: $X \in \mathbf{R}^{n \times d}$, $y \in \mathbf{R}^n$, stepsize sequence $\{\alpha^t\}_{t=1}^{\infty}$, maxiters
Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)
For: $t = 1, \ldots,$ maxiters

- compute gradient $g = X^T(Xw - y)$ ($\mathcal{O}(nd)$ flops)
- update w: ($\mathcal{O}(d)$ flops)

$$w \leftarrow \max(0, w - \alpha^t g)$$

Example: NNLS

$$\text{minimize} \quad \frac{1}{2}\|y - Xw\|^2 + \mathbf{1}(w \geq 0)$$

recall

- $\nabla \left(\frac{1}{2}\|y - Xw\|^2\right) = -X^T(y - Xw)$
- $\mathbf{prox}_{\mathbf{1}(\cdot \geq 0)}(w) = \max(0, w)$

Algorithm Proximal gradient method for NNLS

Given: $X \in \mathbf{R}^{n \times d}$, $y \in \mathbf{R}^n$, stepsize sequence $\{\alpha^t\}_{t=1}^{\infty}$, maxiters
Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)
For: $t = 1, \ldots,$ maxiters

- compute gradient $g = X^T(Xw - y)$ ($\mathcal{O}(nd)$ flops)
- update w: ($\mathcal{O}(d)$ flops)

$$w \leftarrow \max(0, w - \alpha^t g)$$

Example: NNLS

option: do work up front to reduce per-iteration complexity

Algorithm Proximal gradient method for NNLS

Given: $X \in \mathbf{R}^{n \times d}$, $y \in \mathbf{R}^n$, stepsize sequence $\{\alpha^t\}_{t=1}^{\infty}$, maxiters
Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)
compute:

- $b = X^T y$ ($\mathcal{O}(nd)$ flops)
- $G = X^T X$ ($\mathcal{O}(nd^2)$ flops)

For: $t = 1, \ldots,$ maxiters

- compute gradient $g = Gw - b$ ($\mathcal{O}(d^2)$ flops)
- update w: ($\mathcal{O}(d)$ flops)

$$w \leftarrow \max(0, w - \alpha^t g)$$

$\mathcal{O}(nd^2)$ flops to begin, $\mathcal{O}(d^2)$ flops per iteration

Example: Lasso

$$\text{minimize} \quad \frac{1}{2}\|y - Xw\|^2 + \lambda\|w\|_1$$

recall

- $\nabla\left(\frac{1}{2}\|y - Xw\|^2\right) = -X^T(y - Xw)$
- $\mathbf{prox}_{\mu\|\cdot\|_1}(w) = s_\mu(w)$ where

$$(s_\mu(w))_i = \begin{cases} w_i - \mu & w_i \geq \mu \\ 0 & |w_i| \leq \mu \\ w_i + \mu & w_i \leq -\mu \end{cases}$$

Example: Lasso

$$\text{minimize} \quad \frac{1}{2}\|y - Xw\|^2 + \lambda\|w\|_1$$

Algorithm Proximal gradient method for Lasso

Given: $X \in \mathbf{R}^{n\times d}$, $y \in \mathbf{R}^n$, stepsize sequence $\{\alpha^t\}_{t=1}^{\infty}$, maxiters

Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)

For: $t = 1, \ldots,$ maxiters

- compute gradient $g = X^T(Xw - y)$ $\quad$ ($\mathcal{O}(nd)$ flops)
- update w: $\quad$ ($\mathcal{O}(d)$ flops)

$$w \leftarrow s_{\alpha^t\lambda}(w - \alpha^t g)$$

Example: Lasso

$$\text{minimize} \quad \frac{1}{2}\|y - Xw\|^2 + \lambda\|w\|_1$$

Algorithm Proximal gradient method for Lasso

Given: $X \in \mathbf{R}^{n \times d}$, $y \in \mathbf{R}^n$, stepsize sequence $\{\alpha^t\}_{t=1}^{\infty}$, maxiters
Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)
For: $t = 1, \ldots,$ maxiters

- compute gradient $g = X^T(Xw - y)$ ($\mathcal{O}(nd)$ flops)
- update w: ($\mathcal{O}(d)$ flops)

$$w \leftarrow s_{\alpha^t \lambda}(w - \alpha^t g)$$

---

notice: the hard part is computing the gradient!
(can speed this up by precomputation, as for NNLS)

Convergence

two questions to ask:

- will the iteration ever stop?
- what kind of point will it stop at?

if the iteration stops, we say it has **converged**

Convergence: what kind of point will it stop at?

- let's suppose r is differentiable[2]
- if we find w so that

$$w = \mathbf{prox}_{\alpha^t r}(w - \alpha^t \nabla \ell(w))$$

then

$$\begin{aligned} w &= \operatorname*{argmin}_{w'}(\alpha^t r(w') + \frac{1}{2}\|w' - (w - \alpha^t \nabla \ell(w))\|_2^2) \\ 0 &= \nabla \alpha^t r(w) + w - w + \alpha^t \nabla \ell(w) \\ &= \nabla(r(w) + \ell(w)) \end{aligned}$$

- so the gradient of the objective is 0
- if ℓ and r are convex, that means w minimizes $\ell + r$

[2]take Convex Optimization for the proof for non-differentiable r

Convergence: will it stop?

definitions:

- $p^\star = \inf_w \ell(w) + r(w)$

assumptions:

- loss function is continuously differentiable and L-smooth:

$$\ell(w') \leq \ell(w) + \langle \nabla \ell(w), w' - w \rangle + \frac{L}{2}\|w' - w\|^2$$

- for simplicity, consider constant step size $\alpha^t = \alpha$

Proximal point method converges

prove it for $\ell = 0$ first (aka the **proximal point method**)

for any $t = 0, 1, \ldots,$

$$w^{t+1} = \operatorname*{argmin}_w \alpha r(w) + \frac{1}{2}\|w - w^t\|^2$$

so in particular,

$$\begin{aligned} \alpha r(w^{t+1}) + \frac{1}{2}\|w^{t+1} - w^t\|^2 &\leq \alpha r(w^t) + \frac{1}{2}\|w^t - w^t\|^2 \\ \frac{1}{2\alpha}\|w^{t+1} - w^t\|^2 &\leq r(w^t) - r(w^{t+1}) \end{aligned}$$

now add up these inequalities for $t = 0, 1, \ldots, T$:

$$\begin{aligned} \frac{1}{2\alpha}\sum_{t=0}^{T}\|w^{t+1} - w^t\|^2 &\leq \sum_{t=0}^{T}(r(w^t) - r(w^{t+1})) \\ &\leq r(w^0) - p^\star \end{aligned}$$

it converges!

Proximal gradient method converges (I)

now prove it for $\ell \neq 0$. for any $t = 0, 1, \ldots$,

$$w^{t+1} = \operatorname*{argmin}_{w} \alpha r(w) + \frac{1}{2}\|w - (w^t - \alpha \nabla \ell(w^t))\|^2$$

so in particular,

$$\begin{aligned}
&r(w^{t+1}) + \frac{1}{2\alpha}\|w^{t+1} - (w^t - \alpha \nabla \ell(w^t))\|^2 \\
&\qquad \leq r(w^t) + \frac{1}{2\alpha}\|w^t - (w^t - \alpha \nabla \ell(w^t))\|^2 \\
&r(w^{t+1}) + \frac{1}{2\alpha}\|w^{t+1} - w^t\|^2 + \frac{\alpha}{2}\|\nabla \ell(w^t))\|^2 + \langle \nabla \ell(w^t), w^{t+1} - w^t \rangle \\
&\qquad \leq r(w^t) + \frac{\alpha}{2}\|\nabla \ell(w^t)\|^2 \\
&r(w^{t+1}) + \frac{1}{2\alpha}\|w^{t+1} - w^t\|^2 + \langle \nabla \ell(w^t), w^{t+1} - w^t \rangle \\
&\qquad \leq r(w^t)
\end{aligned}$$

Proximal gradient method converges (II)

now use $\ell(w') \le \ell(w) + \langle \nabla \ell(w), w' - w \rangle + \frac{L}{2}\|w' - w\|^2$
with $w' = w^{t+1}$, $w = w^t$

$$\ell(w^{t+1}) + r(w^{t+1}) + \frac{1}{2\alpha}\|w^{t+1} - w^t\|^2 + \langle \nabla \ell(w^t), w^{t+1} - w^t \rangle$$
$$\le \ell(w^t) + r(w^t) + \langle \nabla \ell(w^t), w^{t+1} - w^t \rangle + \frac{L}{2}\|w^{t+1} - w^t\|^2$$

$$\ell(w^{t+1}) + r(w^{t+1}) + \frac{1}{2\alpha}\|w^{t+1} - w^t\|^2$$
$$\le \ell(w^t) + r(w^t) + \frac{L}{2}\|w^{t+1} - w^t\|^2$$

$$\frac{1}{2\alpha}\|w^{t+1} - w^t\|^2 - \frac{L}{2}\|w^{t+1} - w^t\|^2$$
$$\le \ell(w^t) + r(w^t) - (\ell(w^{t+1}) + r(w^{t+1}))$$

Proximal gradient method converges (III)

now add up these inequalities for $t = 0, 1, \ldots, T$:

$$\begin{aligned} \frac{1}{2}\left(\frac{1}{\alpha} - L\right) \sum_{t=0}^{T} \|w^{t+1} - w^t\|^2 \\ &\leq \sum_{t=0}^{T} \ell(w^t) + r(w^t) - (\ell(w^{t+1}) + r(w^{t+1}))) \\ &\leq \ell(w^0) + r(w^0) - p^\star \end{aligned}$$

if

$$\begin{aligned} \frac{1}{\alpha} - L &\geq 0 \\ \implies \alpha &\leq \frac{1}{L} \end{aligned}$$

it converges!

Stochastic proximal subgradient

Algorithm Stochastic proximal subgradient method

Given: loss $\ell : \mathbf{R}^d \to \mathbf{R}$, regularizer $r : \mathbf{R}^d \to \mathbf{R}$, stepsizes $\{\alpha^t\}_{t=1}^{\infty}$, maxiters

Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)

For: $t = 1, \ldots,$ maxiters

- pick $S \subseteq \{1, \ldots, n\}$ uniformly at random
- pick $g \in \tilde{\partial}\ell(w)$ ($\mathcal{O}(|S|d)$ flops)
- update w: ($\mathcal{O}(d)$ flops)

$$w \leftarrow \mathbf{prox}_{\alpha^t r}(w - \alpha^t g)$$

Stochastic proximal subgradient

Algorithm Stochastic proximal subgradient method

Given: loss $\ell : \mathbf{R}^d \to \mathbf{R}$, regularizer $r : \mathbf{R}^d \to \mathbf{R}$, stepsizes $\{\alpha^t\}_{t=1}^{\infty}$, maxiters
Initialize: $w \in \mathbf{R}^d$ (often, $w = 0$)
For: $t = 1, \ldots,$ maxiters

- pick $S \subseteq \{1, \ldots, n\}$ uniformly at random
- pick $g \in \tilde{\partial}\ell(w)$ ($\mathcal{O}(|S|d)$ flops)
- update w: ($\mathcal{O}(d)$ flops)

$$w \leftarrow \mathbf{prox}_{\alpha^t r}(w - \alpha^t g)$$

per iteration complexity: $\mathcal{O}(|S|d)$

Convergence for stochastic proximal (sub)gradient

pick your poison:

- stochastic (sub)gradient, fixed step size $\alpha^t = \alpha$:
  - iterates converge quickly, then wander within a small ball
- stochastic (sub)gradient, decreasing step size $\alpha^t = 1/t$:
  - iterates converge slowly to solution
- minibatch stochastic (sub)gradient with increasing minibatch size, fixed step size $\alpha^t = \alpha$:
  - iterates converge quickly to solution
  - later iterations take (much) longer

proofs: [Bertsekas, 2010] `https://arxiv.org/pdf/1507.01030v1.pdf`

conditions:

- ℓ is convex, subdifferentiable, Lipschitz continuous, and
- r is convex and Lipschitz continuous where it is $< \infty$

or

- all iterates are bounded

References

- Beck: book chapter on proximal operators (lots of examples!) https://archive.siam.org/books/mo25/mo25_ch6.pdf
- Vandenberghe: lecture on proximal gradient method. http://www.seas.ucla.edu/~vandenbe/236C/lectures/proxgrad.pdf
- Yin: lecture on proximal method. http://www.math.ucla.edu/~wotaoyin/summer2013/slides/Lec05_ProximalOperatorDual.pdf
- Parikh and Boyd: paper on proximal algorithms. https://stanford.edu/~boyd/papers/pdf/prox_algs.pdf
- Bertsekas: convergence proofs for every proximal gradient style method you can dream of. https://arxiv.org/pdf/1507.01030v1.pdf