Convex Optimization

(EE227A: UC Berkeley)

Lecture 18
(Proximal methods; Incremental methods – I)

21 March, 2013

Suvrit Sra

Douglas-Rachford method

$$\boxed{0 \in \partial f(x) + \partial g(x)}$$

Douglas-Rachford method

$$\boxed{0 \in \partial f(x) + \partial g(x)}$$

DR method: given z^0, iterate for $k \geq 0$

$$\begin{aligned} x^k &= \mathrm{prox}_g(z^k) \\ v^k &= \mathrm{prox}_f(2x^k - z^k) \\ z^{k+1} &= z^k + \gamma_k(v^k - x^k) \end{aligned}$$

Douglas-Rachford method

$$\boxed{0 \in \partial f(x) + \partial g(x)}$$

DR method: given z^0, iterate for $k \geq 0$

$$\begin{aligned} x^k &= \mathrm{prox}_g(z^k) \\ v^k &= \mathrm{prox}_f(2x^k - z^k) \\ z^{k+1} &= z^k + \gamma_k(v^k - x^k) \end{aligned}$$

For $\gamma_k = 1$, we have

$$z^{k+1} = z^k + v^k - x^k$$
$$z^{k+1} = z^k + \mathrm{prox}_f(2\,\mathrm{prox}_g(z^k) - z^k) - \mathrm{prox}_g(z^k)$$

Douglas-Rachford method

$$z^{k+1} = z^k + \operatorname{prox}_f(2\operatorname{prox}_g(z^k) - z^k) - \operatorname{prox}_g(z^k)$$

Douglas-Rachford method

$$z^{k+1} = z^k + \mathrm{prox}_f(2\,\mathrm{prox}_g(z^k) - z^k) - \mathrm{prox}_g(z^k)$$

Dropping superscripts, we have the fixed-point iteration

$$\boxed{\begin{gathered} z \leftarrow Tz \\ T = I + P_f(2P_g - I) - P_g \end{gathered}}$$

Douglas-Rachford method

$$z^{k+1} = z^k + \operatorname{prox}_f(2\operatorname{prox}_g(z^k) - z^k) - \operatorname{prox}_g(z^k)$$

Dropping superscripts, we have the fixed-point iteration

$$\boxed{\begin{gathered} z \leftarrow Tz \\ T = I + P_f(2P_g - I) - P_g \end{gathered}}$$

Lemma DR can be written as: $z \leftarrow \frac{1}{2}(R_f R_g + I)z$, where R_f denotes the *reflection operator* $2P_f - I$ (similarly R_g).

Exercise: Prove this claim.

Proximity for several functions

Optimizing sums of functions

$$
\begin{aligned}
f(x) &:= \tfrac{1}{2}\|x-y\|_2^2 + \sum_i f_i(x) \\
f(x) &:= \sum_i f_i(x)
\end{aligned}
$$

Proximity for several functions

Optimizing sums of functions

$$
\begin{aligned}
f(x) &:= \tfrac{1}{2}\|x-y\|_2^2 + \sum\nolimits_i f_i(x) \\
f(x) &:= \sum\nolimits_i f_i(x)
\end{aligned}
$$

$$\boxed{\text{DR does not work immediately}}$$

Product space trick

- Original problem over $\mathcal{H} = \mathbb{R}^n$

Product space trick

- Original problem over $\mathcal{H} = \mathbb{R}^n$
- Suppose we have $\sum_{i=1}^m f_i(x)$

Product space trick

- Original problem over $\mathcal{H} = \mathbb{R}^n$
- Suppose we have $\sum_{i=1}^{m} f_i(x)$
- Introduce new variables $(x_1, \ldots, x_m)$

Product space trick

- Original problem over $\mathcal{H} = \mathbb{R}^n$
- Suppose we have $\sum_{i=1}^m f_i(x)$
- Introduce new variables $(x_1, \ldots, x_m)$
- Now problem is over domain $\mathcal{H}^m := \mathcal{H} \times \mathcal{H} \times \cdots \times \mathcal{H}$ (m-times)

Product space trick

- Original problem over $\mathcal{H} = \mathbb{R}^n$
- Suppose we have $\sum_{i=1}^m f_i(x)$
- Introduce new variables $(x_1, \ldots, x_m)$
- Now problem is over domain $\mathcal{H}^m := \mathcal{H} \times \mathcal{H} \times \cdots \times \mathcal{H}$ (m-times)
- New constraint: $x_1 = x_2 = \ldots = x_m$

$$\begin{aligned} \min_{(x_1, \ldots, x_m)} \quad & \sum\nolimits_i f_i(x_i) \\ \text{s.t.} \quad & x_1 = x_2 = \cdots = x_m. \end{aligned}$$

Product space trick

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \mathbb{I}_{\mathcal{B}}(\boldsymbol{x})$$

where $\boldsymbol{x} \in \mathcal{H}^m$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathcal{H}^m \mid \boldsymbol{z} = (x, x, \ldots, x)\}$

Product space trick

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \mathbb{I}_{\mathcal{B}}(\boldsymbol{x})$$

where $\boldsymbol{x} \in \mathcal{H}^m$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathcal{H}^m \mid \boldsymbol{z} = (x, x, \ldots, x)\}$

- Let $\boldsymbol{y} = (y_1, \ldots, y_m)$

Product space trick

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \mathbb{I}_{\mathcal{B}}(\boldsymbol{x})$$

where $\boldsymbol{x} \in \mathcal{H}^m$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathcal{H}^m \mid \boldsymbol{z} = (x, x, \ldots, x)\}$

- Let $\boldsymbol{y} = (y_1, \ldots, y_m)$
- $\mathrm{prox}_f(\boldsymbol{y}) = (\mathrm{prox}_{f_1}(y_1), \ldots, \mathrm{prox}_{f_m}(y_m))$

Product space trick

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \mathbb{I}_{\mathcal{B}}(\boldsymbol{x})$$

where $\boldsymbol{x} \in \mathcal{H}^m$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathcal{H}^m \mid \boldsymbol{z} = (x, x, \ldots, x)\}$

- Let $\boldsymbol{y} = (y_1, \ldots, y_m)$
- $\operatorname{prox}_f(\boldsymbol{y}) = (\operatorname{prox}_{f_1}(y_1), \ldots, \operatorname{prox}_{f_m}(y_m))$
- $P_{\mathcal{B}}(\boldsymbol{y})$ can be solved as follows:

Product space trick

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \mathbb{I}_{\mathcal{B}}(\boldsymbol{x})$$

where $\boldsymbol{x} \in \mathcal{H}^m$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathcal{H}^m \mid \boldsymbol{z} = (x, x, \ldots, x)\}$

- Let $\boldsymbol{y} = (y_1, \ldots, y_m)$
- $\mathrm{prox}_f(\boldsymbol{y}) = (\mathrm{prox}_{f_1}(y_1), \ldots, \mathrm{prox}_{f_m}(y_m))$
- $P_{\mathcal{B}}(\boldsymbol{y})$ can be solved as follows:

$$\begin{array}{cc} \min_{\boldsymbol{z} \in \mathcal{B}} & \frac{1}{2}\|\boldsymbol{z} - \boldsymbol{y}\|_2^2 \\ \min_{x \in \mathcal{H}} & \sum_i \frac{1}{2}\|x - y_i\|_2^2 \\ \Longrightarrow & x = \frac{1}{m}\sum_i y_i \end{array}$$

Exercise: Work out the details of DR with the above ideas.

Note: this trick works for all other situations!

Proximity operator for sums

$$\boxed{\min_x \ \tfrac{1}{2}\|x - y\|_2^2 + g(x) + h(x)}$$

Proximity operator for sums

$$\min_x \ \frac{1}{2}\|x - y\|_2^2 + g(x) + h(x)$$

Usually $\mathrm{prox}_{f+g} \neq \mathrm{prox}_f \circ \mathrm{prox}_g$

Proximity operator for sums

$$\min_x \ \tfrac{1}{2}\|x - y\|_2^2 + g(x) + h(x)$$

Usually $\operatorname{prox}_{f+g} \neq \operatorname{prox}_f \circ \operatorname{prox}_g$

Proximal-Dykstra method

1. Let $x^0 = y$; $u^0 = 0$, $z^0 = 0$
2. k-th iteration ($k \geq 0$)

Proximity operator for sums

$$\min_x \ \tfrac{1}{2}\|x-y\|_2^2 + g(x) + h(x)$$

Usually $\mathrm{prox}_{f+g} \neq \mathrm{prox}_f \circ \mathrm{prox}_g$

Proximal-Dykstra method

1. Let $x^0 = y$; $u^0 = 0$, $z^0 = 0$
2. k-th iteration ($k \geq 0$)
   - $w^k = \mathrm{prox}_g(x^k + z^k)$
   - $u^{k+1} = x^k + u^k - w^k$

Proximity operator for sums

$$\min_x \ \tfrac{1}{2}\|x-y\|_2^2 + g(x) + h(x)$$

Usually $\text{prox}_{f+g} \neq \text{prox}_f \circ \text{prox}_g$

Proximal-Dykstra method

1. Let $x^0 = y$; $u^0 = 0$, $z^0 = 0$
2. k-th iteration ($k \geq 0$)
   - $w^k = \text{prox}_g(x^k + z^k)$
   - $u^{k+1} = x^k + u^k - w^k$
   - $x^{k+1} = \text{prox}_h(w^k + z^k)$
   - $z^{k+1} = w^k + z^k - x^{k+1}$

Proximity operator for sums

$$\min_x \ \frac{1}{2}\|x-y\|_2^2 + g(x) + h(x)$$

Usually $\mathrm{prox}_{f+g} \neq \mathrm{prox}_f \circ \mathrm{prox}_g$

Proximal-Dykstra method

1. Let $x^0 = y$; $u^0 = 0$, $z^0 = 0$
2. k-th iteration ($k \geq 0$)
   - $w^k = \mathrm{prox}_g(x^k + z^k)$
   - $u^{k+1} = x^k + u^k - w^k$
   - $x^{k+1} = \mathrm{prox}_h(w^k + z^k)$
   - $z^{k+1} = w^k + z^k - x^{k+1}$

Why does it work?

Proximity operator for sums

$$\min_x \ \frac{1}{2}\|x-y\|_2^2 + g(x) + h(x)$$

Usually $\mathrm{prox}_{f+g} \neq \mathrm{prox}_f \circ \mathrm{prox}_g$

Proximal-Dykstra method

1. Let $x^0 = y$; $u^0 = 0$, $z^0 = 0$
2. k-th iteration ($k \geq 0$)
   - $w^k = \mathrm{prox}_g(x^k + z^k)$
   - $u^{k+1} = x^k + u^k - w^k$
   - $x^{k+1} = \mathrm{prox}_h(w^k + z^k)$
   - $z^{k+1} = w^k + z^k - x^{k+1}$

Why does it work? After the break...!

Proximity operator for sums

$$\min_x \;\; \tfrac{1}{2}\|x - y\|_2^2 + g(x) + h(x)$$

Usually $\mathrm{prox}_{f+g} \neq \mathrm{prox}_f \circ \mathrm{prox}_g$

Proximal-Dykstra method

1. Let $x^0 = y$; $u^0 = 0$, $z^0 = 0$
2. k-th iteration ($k \geq 0$)
   - $w^k = \mathrm{prox}_g(x^k + z^k)$
   - $u^{k+1} = x^k + u^k - w^k$
   - $x^{k+1} = \mathrm{prox}_h(w^k + z^k)$
   - $z^{k+1} = w^k + z^k - x^{k+1}$

Why does it work? After the break...!

Exercise: Use the product-space trick to extend this to a *parallel Dykstra-like* method for $m \geq 3$ functions.

Incremental methods

Separable objectives

$$\boxed{\min \quad f(x) = \sum_i^m f_i(x) + \lambda r(x)}$$

Separable objectives

$$\boxed{\min \quad f(x) = \sum_i^m f_i(x) + \lambda r(x)}$$

Gradient / subgradient methods

$$\begin{aligned}
x^{k+1} &= x^k - \alpha_k \nabla f(x^k) \qquad \lambda = 0, \\
x^{k+1} &= x_k - \alpha_k g(x^k), \qquad g(x^k) \in \partial f(x^k) + \lambda \partial r(x^k) \\
x^{k+1} &= \mathrm{prox}_{\alpha_k r}(x^k - \alpha_k \nabla f(x^k))
\end{aligned}$$

Separable objectives

$$\boxed{\min \quad f(x) = \sum_i^m f_i(x) + \lambda r(x)}$$

Gradient / subgradient methods

$$\begin{aligned}
x^{k+1} &= x^k - \alpha_k \nabla f(x^k) \qquad \lambda = 0, \\
x^{k+1} &= x_k - \alpha_k g(x^k), \qquad g(x^k) \in \partial f(x^k) + \lambda \partial r(x^k) \\
x^{k+1} &= \mathrm{prox}_{\alpha_k r}(x^k - \alpha_k \nabla f(x^k))
\end{aligned}$$

How much computation does one iteration take?

Incremental gradient methods

What if at iteration k, we randomly pick an integer $i(k) \in \{1, 2, \ldots, m\}$?

Incremental gradient methods

What if at iteration k, we randomly pick an integer $i(k) \in \{1, 2, \ldots, m\}$?

And instead just perform the update?

$$x^{k+1} = x^k - \alpha_k \nabla f_{i(k)}(x_k)$$

Incremental gradient methods

What if at iteration k, we randomly pick an integer $i(k) \in \{1, 2, \ldots, m\}$?

And instead just perform the update?

$$x^{k+1} = x^k - \alpha_k \nabla f_{i(k)}(x_k)$$

- The update requires only gradient for $f_{i(k)}$
- One iteration now m times faster than with $\nabla f(x)$

Incremental gradient methods

What if at iteration k, we randomly pick an integer $i(k) \in \{1, 2, \ldots, m\}$?

And instead just perform the update?

$$x^{k+1} = x^k - \alpha_k \nabla f_{i(k)}(x_k)$$

- The update requires only gradient for $f_{i(k)}$
- One iteration now m times faster than with $\nabla f(x)$

But does this make sense?

Incremental gradient methods

♡ Old idea; has been used extensively as *backpropagation* in neural networks, Widrow-Hoff least mean squares, gradient methods with errors, stochastic gradient, etc.

Incremental gradient methods

- ♡ Old idea; has been used extensively as *backpropagation* in neural networks, Widrow-Hoff least mean squares, gradient methods with errors, stochastic gradient, etc.
- ♡ Can effectively use to "stream" through data — go through components one by one, say *cyclically* instead of randomly

Incremental gradient methods

- ♡ Old idea; has been used extensively as *backpropagation* in neural networks, Widrow-Hoff least mean squares, gradient methods with errors, stochastic gradient, etc.
- ♡ Can effectively use to "stream" through data — go through components one by one, say *cyclically* instead of randomly
- ♡ If m is very large, many of the $f_i(x)$ may have similar minimizers;

Incremental gradient methods

- ♡ Old idea; has been used extensively as *backpropagation* in neural networks, Widrow-Hoff least mean squares, gradient methods with errors, stochastic gradient, etc.
- ♡ Can effectively use to "stream" through data — go through components one by one, say *cyclically* instead of randomly
- ♡ If m is very large, many of the $f_i(x)$ may have similar minimizers; by using the f_i only individually we hope to take advantage of this fact, and greatly speed up.

Incremental gradient methods

- ♡ Old idea; has been used extensively as *backpropagation* in neural networks, Widrow-Hoff least mean squares, gradient methods with errors, stochastic gradient, etc.
- ♡ Can effectively use to "stream" through data — go through components one by one, say *cyclically* instead of randomly
- ♡ If m is very large, many of the $f_i(x)$ may have similar minimizers; by using the f_i only individually we hope to take advantage of this fact, and greatly speed up.
- ♡ Incremental methods usually effective far from the eventual limit (solution) — become very slow close to the solution.

Incremental gradient methods

- ♡ Old idea; has been used extensively as *backpropagation* in neural networks, Widrow-Hoff least mean squares, gradient methods with errors, stochastic gradient, etc.
- ♡ Can effectively use to "stream" through data — go through components one by one, say *cyclically* instead of randomly
- ♡ If m is very large, many of the $f_i(x)$ may have similar minimizers; by using the f_i only individually we hope to take advantage of this fact, and greatly speed up.
- ♡ Incremental methods usually effective far from the eventual limit (solution) — become very slow close to the solution.

Example!

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Solving $f'(x) = 0$ we obtain

$$x^* = \frac{\sum_i a_i b_i}{\sum_i a_i^2}$$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Solving $f'(x) = 0$ we obtain

$$x^* = \frac{\sum_i a_i b_i}{\sum_i a_i^2}$$

- Minimum of a single $f_i(x) = \frac{1}{2}(a_i x - b_i)^2$ is $x_i^* = b_i/a_i$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \frac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Solving $f'(x) = 0$ we obtain

$$x^* = \frac{\sum_i a_i b_i}{\sum_i a_i^2}$$

- Minimum of a single $f_i(x) = \frac{1}{2}(a_i x - b_i)^2$ is $x_i^* = b_i/a_i$
- Notice now that

$$x^* \in [\min_i x_i^*, \max_i x_i^*] =: R$$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Notice: $x^* \in [\min_i x_i^*, \max_i x_i^*] =: R$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum\nolimits_{i=1}^{m}(a_i x - b_i)^2$$

- Notice: $x^* \in [\min_i x_i^*, \max_i x_i^*] =: R$
- If we have a scalar x that lies outside R?
- We see that

$$\nabla f_i(x) = a_i(a_i x - b_i)$$
$$\nabla f(x) = \sum\nolimits_i a_i(a_i x - b_i)$$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Notice: $x^* \in [\min_i x_i^*, \max_i x_i^*] =: R$
- If we have a scalar x that lies outside R?
- We see that

$$\nabla f_i(x) = a_i(a_i x - b_i)$$
$$\nabla f(x) = \sum_i a_i(a_i x - b_i)$$

- $\nabla f_i(x)$ has **same sign** as $\nabla f(x)$ So using $\nabla f_i(x)$ **instead** of $\nabla f(x)$ also ensures progress.

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \frac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Notice: $x^* \in [\min_i x_i^*, \max_i x_i^*] =: R$
- If we have a scalar x that lies outside R?
- We see that

$$\nabla f_i(x) = a_i(a_i x - b_i)$$
$$\nabla f(x) = \sum_i a_i(a_i x - b_i)$$

- $\nabla f_i(x)$ has **same sign** as $\nabla f(x)$ So using $\nabla f_i(x)$ **instead** of $\nabla f(x)$ also ensures progress.
- But once inside region R, **no guarantee** that incremental method will make progress towards optimum.

Incremental proximal method

$$\boxed{\min \quad f(x) = \sum_i f_i(x)}$$

What if the f_i are nonsmooth?

Incremental proximal method

$$\boxed{\min \quad f(x) = \sum_i f_i(x)}$$

What if the f_i are nonsmooth?

$$\cancel{x^{k+1} = \operatorname{prox}_{\alpha_k f}(x^k)}$$

Incremental proximal method

$$\boxed{\min \quad f(x) = \sum_i f_i(x)}$$

What if the f_i are nonsmooth?

$$\cancel{x^{k+1} = \operatorname{prox}_{\alpha_k f}(x^k)}$$

$$x^{k+1} = \operatorname{prox}_{\alpha_k f_{i(k)}}(x^k)$$

$$x^{k+1} = \operatorname{argmin}\left(\tfrac{1}{2}\|x - x_k\|_2^2 + f_{i(k)}(x)\right)$$

$i(k) \in \{1, 2, \ldots, m\}$ picked uniformly at random.

Incremental proximal-gradients

$$\min \quad \sum_i f_i(x) + r(x).$$

Incremental proximal-gradients

$$\min \quad \sum_i f_i(x) + r(x).$$

$$x^{k+1} \quad = \quad \operatorname{prox}_{\eta_k r}\big(x^k - \eta_k \sum_{i=1}^m \nabla f_i(z^i)\big), \quad k = 0, 1, \ldots,$$

Incremental proximal-gradients

$$\min \quad \sum_i f_i(x) + r(x).$$

$$\begin{aligned} x^{k+1} \quad &= \quad \operatorname{prox}_{\eta_k r}\Big(x^k - \eta_k \sum_{i=1}^m \nabla f_i(z^i)\Big), \quad k = 0, 1, \ldots, \\ & \quad\;\; z^1 = x^k \\ & \quad\;\; z^{i+1} = z^i - \eta_k \nabla f_i(z^i), \quad i = 1, \ldots, m-1. \end{aligned}$$

Incremental proximal-gradients

$$\min \quad \sum_i f_i(x) + r(x).$$

$$\begin{aligned} x^{k+1} &= \operatorname{prox}_{\eta_k r}\Big(x^k - \eta_k \sum_{i=1}^m \nabla f_i(z^i)\Big), \quad k = 0, 1, \ldots, \\ & z^1 = x^k \\ & z^{i+1} = z^i - \eta_k \nabla f_i(z^i), \quad i = 1, \ldots, m-1. \end{aligned}$$

We can choose $\eta_k = 1/L$, where L is Lipschitz constant of $\nabla f(x)$

Incremental proximal-gradients

$$\min \quad \sum_i f_i(x) + r(x).$$

$$\begin{aligned} x^{k+1} \quad &= \quad \operatorname{prox}_{\eta_k r}\Big(x^k - \eta_k \sum_{i=1}^{m} \nabla f_i(z^i)\Big), \quad k = 0, 1, \ldots, \\ & \quad\quad z^1 = x^k \\ & \quad\quad z^{i+1} = z^i - \eta_k \nabla f_i(z^i), \quad i = 1, \ldots, m-1. \end{aligned}$$

We can choose $\eta_k = 1/L$, where L is Lipschitz constant of $\nabla f(x)$

$\boxed{\text{Does this work?}}$

Incremental methods: key realization

$$\min \quad (f(x) = \sum_i f_i(x)) + r(x)$$

Gradient with error

$$\nabla f_{i(k)}(x) = \nabla f(x) + e(x)$$
$$x^{k+1} = \mathrm{prox}_{\alpha r}[x^k - \alpha_k(\nabla f(x^k) + e(x^k))]$$

Incremental methods: key realization

$$\min \quad (f(x) = \sum_i f_i(x)) + r(x)$$

Gradient with error

$$\nabla f_{i(k)}(x) = \nabla f(x) + e(x)$$
$$x^{k+1} = \mathrm{prox}_{\alpha r}[x^k - \alpha_k(\nabla f(x^k) + e(x^k))]$$

So if in the limit error $\alpha_k e(x^k)$ disappears, we should be ok!

Incremental gradient methods

Incremental gradient methods may be viewed as

Gradient methods with error in gradient computation

Incremental gradient methods

Incremental gradient methods may be viewed as

Gradient methods with error in gradient computation

- If we can control this error, we can control convergence

Incremental gradient methods

Incremental gradient methods may be viewed as

Gradient methods with error in gradient computation

- If we can control this error, we can control convergence
- Error makes even smooth case more like nonsmooth case

Incremental gradient methods

Incremental gradient methods may be viewed as

Gradient methods with error in gradient computation

- If we can control this error, we can control convergence
- Error makes even smooth case more like nonsmooth case
- So, convergence crucially depends on stepsize α_k

Incremental gradient methods

Incremental gradient methods may be viewed as

Gradient methods with error in gradient computation

- If we can control this error, we can control convergence
- Error makes even smooth case more like nonsmooth case
- So, convergence crucially depends on stepsize α_k

Some stepsize choices

♠ $\alpha_k = c$, a small enough constant

♠ $\alpha_k \to 0$, $\sum_k \alpha_k = \infty$ (diminishing scalar)

♠ Constant for some iterations, diminish, again constant, repeat

♠ $\alpha_k = \min(c, a/(b+k))$, where $a, b, c > 0$ (user chosen).

Incremental gradient – summary

♠ Usually much faster (large m) when *far* from convergence

Incremental gradient – summary

- ♠ Usually much faster (large m) when *far* from convergence
- ♠ Slow progress near optimum (because α_k often too small)

Incremental gradient – summary

- ♠ Usually much faster (large m) when *far* from convergence
- ♠ Slow progress near optimum (because α_k often too small)
- ♠ Constant step $\alpha_k = \alpha$, doesn't always yield convergence

Incremental gradient – summary

- Usually much faster (large m) when *far* from convergence
- Slow progress near optimum (because α_k often too small)
- Constant step $\alpha_k = \alpha$, doesn't always yield convergence
- Diminishing step $\alpha_k = O(1/k)$ leads to convergence

Incremental gradient – summary

- ♠ Usually much faster (large m) when *far* from convergence
- ♠ Slow progress near optimum (because α_k often too small)
- ♠ Constant step $\alpha_k = \alpha$, doesn't always yield convergence
- ♠ Diminishing step $\alpha_k = O(1/k)$ leads to convergence
- ♠ Slow, sublinear rate of convergence

Incremental gradient – summary

- ♠ Usually much faster (large m) when *far* from convergence
- ♠ Slow progress near optimum (because α_k often too small)
- ♠ Constant step $\alpha_k = \alpha$, doesn't always yield convergence
- ♠ Diminishing step $\alpha_k = O(1/k)$ leads to convergence
- ♠ Slow, sublinear rate of convergence
- ♠ Optimal, incremental method seems not to be known

Incremental gradient – summary

- ♠ Usually much faster (large m) when *far* from convergence
- ♠ Slow progress near optimum (because α_k often too small)
- ♠ Constant step $\alpha_k = \alpha$, doesn't always yield convergence
- ♠ Diminishing step $\alpha_k = O(1/k)$ leads to convergence
- ♠ Slow, sublinear rate of convergence
- ♠ Optimal, incremental method seems not to be known
- ♠ Idea extends to subgradient, and proximal setups

Incremental gradient – summary

- Usually much faster (large m) when *far* from convergence
- Slow progress near optimum (because α_k often too small)
- Constant step $\alpha_k = \alpha$, doesn't always yield convergence
- Diminishing step $\alpha_k = O(1/k)$ leads to convergence
- Slow, sublinear rate of convergence
- Optimal, incremental method seems not to be known
- Idea extends to subgradient, and proximal setups
- Some extensions also apply to nonconvex problems

Incremental gradient – summary

- ♠ Usually much faster (large m) when *far* from convergence
- ♠ Slow progress near optimum (because α_k often too small)
- ♠ Constant step $\alpha_k = \alpha$, doesn't always yield convergence
- ♠ Diminishing step $\alpha_k = O(1/k)$ leads to convergence
- ♠ Slow, sublinear rate of convergence
- ♠ Optimal, incremental method seems not to be known
- ♠ Idea extends to subgradient, and proximal setups
- ♠ Some extensions also apply to nonconvex problems
- ♠ Some extend to parallel and distributed computation

Read (omit proofs): "Incremental methods survey" by D. P. Bertsekas (2010) – see bSpace.

References

1. Combettes and Pesquet. *Proximal splitting methods in signal processing.* (2010)
2. Bertsekas. *Nonlinear Programming.* (1999).