Proximal Algorithm

6:57 PM

* Now let's switch to Boyd's proximal gradient paper:

Before proceeding any further let's look at the difference in notation between Calafiore and Boyd

Calafiore	Boyd
$f_0(x)$	$f(x)$
$h(x)$	$g(x)$
s_k	λ^k
x_{k+1}	x^k
index starts with 0	index starts at 1
$x_{k+1} = \mathrm{prox}_{s_k h}(x_k - s_k \nabla f_0(x_k))$	$x^{k+1} = \mathrm{prox}_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$

```
Proximal_Gradient_Method_with_variable_step_size (f :: DifferentiableFunction, g :: NondifferentiableFunction, ε :: Tolerance)
x^1 = 0 # say
λ^0 = 1e-5 # say
k = 1
β = 0.5 # β ∈ (0,1)
while(1)
λ^k, x^{k+1} = BT_ls_nxt_pt_gntr(x^k, λ^{k-1}, β)
if ||x^{k+1} - x^k||_2 ≤ ε
    break
else
    k := k+1
end # if
return x^k
end # while
```

remember proximal gradient
algorithm is a fixed point problem
i.e $x^* = \mathrm{prox}_{\lambda g}(x^* - \lambda \nabla f_0(x^*))$
that is why terminating condition
is of the form $\|x^{k+1} - x^k\|_2 \leq \varepsilon$

```
function BT_ls_nxt_pt_gntr(x^k :: vector, λ^{k-1} :: scalar, β :: scalar)      β ∈ (0,1)
# Beck Teboulle line search and next point generator
λ := λ^{k-1}        # λ^{k-1} denotes x^k = prox_{λ^{k-1} g}(x^{k-1} - λ^{k-1} ∇f(x^k)) which we already know from given
while(1)
    z := prox_{λg}(x^k - λ∇f(x^k))   # fixed stepsize (as in Calafiore) would set x^{k+1} = z right away, however as in this
                                      # step size is not fixed
    if (f(z) ≤ f̂_λ(z, x^k))  # f̂_λ(x,y) = f(y) + ∇f(y)^T (x-y) + 1/(2λ) ||x-y||_2^2 : this is an upper bound for f(x)
        break
    else λ := βλ
    end # if
end # while
```

end # while
end # function

is of the form $\|x^{k+1}-x^{k}\|_2 \leq \varepsilon$

end # if
end # while
return $\lambda^k := \lambda$, $x^{k+1} := z$ # then, $x^{k+1} = \text{prox}_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$
end # function

4.1. Proximal minimization # also known as proximal iteration, proximal point algorithm

$x^{k+1} := \text{prox}_{\lambda f}(x^k)$

$\therefore \mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$, closed proper convex

- Convergence guranteed for $\lambda^k > 0$, $\sum_{k=1}^{\infty} \lambda^k = \infty$

Moreau-Yosida regularization, M_f

- Interpretation: 1) gradient method applied to Moreau envelope M_f

because, $\text{prox}_{\lambda f}(x^k) = x^k - \lambda \nabla M_{\lambda f}(x^k)$ # $\text{prox}_{\lambda f}(x) = x - \lambda \nabla M_{\lambda f}(x)$

$\therefore x^{k+1} = x^k - \lambda \nabla M_{\lambda f}(x^k)$

2) Simple iteration to find fixed point of $\text{prox}_{\lambda f}(x)$

4.2. Proximal gradient method: (for detailed convergence proof see Proximal algorithm Calafiore

$\min f(x) + g(x)$

$f: \mathbb{R}^n \to \mathbb{R}$, CPC

$g: \mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$, CPC

can encode constraints on variable x for being extended valued

- The objective is split into two terms

nonunique splitting → different splitting leads to different nonunique splitting

∇f : Lipschitz continuous $\Leftrightarrow \forall_{x,y}\ \|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x-y\|$

% this is a way of bounding the hessian
so in the taylor series the effect of second order terms will be negligible

different nonunique splitting

∇f : Lipschitz continuous $\Leftrightarrow \forall_{x,y} \; \|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x-y\|$

% this is a way of bounding the hessian
so in the taylor series the effect of second order terms will be negligible

Proximal gradient algorithm:

$$x^{k+1} = \text{prox}_{\lambda^k g}\left(x^k - \lambda^k \nabla f(x^k)\right)$$

$\lambda^k > 0$

Theoretical results: (∇f: Lipschitz continuous, L Lipschitz constant,

$\lambda^k = \lambda$: fixed stepsize, $\in (0, 1/L]$ $\rightarrow$ converges with $O\left(\frac{1}{k}\right)$)

- L not known (most of the time): use Beck-Teboulle proximal gradient updater

Beck-Teboulle proximal gradient updater

Because, $x^k = \text{prox}_{\lambda^{k-1} g}\left(x^{k-1} - \lambda \nabla f(x^k)\right)$

```
Beck-Teboulle_prox_gradient_updater(x^k, λ^(k-1), β)   # will output x^(k+1), and λ^k for which x^(k+1) = prox_{λ^k g}(x^k - λ∇f(x^k))
                                          β ∈ (0,1)
λ := λ^(k-1)
while(1)
  z := prox_{λg}(x^k - λ∇f(x^k))
  if f(z) ≤ f̂_λ(x, x^k)   # f̂_λ(x, x^k) = f(y) + ∇f(y)^T (x - y) + 1/(2λ) ||x - y||_2^2 : a valid upper bound for f
    return λ^k = λ, x^(k+1) = z
  end
  λ = βλ
end
```

z is a candidate for the x^(k+1), however our \lambda here is still the \lamda^(k-1) of the previous step, so what we do is keep making \lambda smaller until it satisfies the Beck-Teboulle line search condition f(z) \leq \hat{f_\lambda}(x,x^k)

A justification of this coming soon [#Intuitive Explanation behind Beck line search condition]

In short we are checking if $\hat{f}_\lambda(x, x^k)$ can act as a valid majorization, (From definition it is convex, $\hat{f}_\lambda(x, x) = f(x)$, all we have to check is the upperboundedness

[def: majorization]

If we have arrived at this line, then for sure the if Beck-Teboulle line search condition was not met i.e. for that \lambda the upperboundedness condition is not met
, so we need to make \lambda smaller and then check the line search condition again. Essentially we keep reducing \lambda i.e., increasing 1/ 2 \lambda until, $\hat{f}_\lambda(x, x^k)$ becomes a majorization at x^k

Evolution of subgradient method to proximal gradient method:

normal subgradient

$$x^{k+1} = x^k - \lambda^k \underbrace{g^k}_{g^k \in \partial f(x^k)} \quad : \text{solves } \underset{}{\min}\, f(x)$$

projected subgradient

$$x^{k+1} = \left[x^k - \lambda^k g^k\right]_X = \Pi_X\left(x^k - \lambda^k g^k\right) \; : \text{solves} \begin{pmatrix} \min f(x) \\ x \in X \end{pmatrix} = \min f(x) + I_X(x)$$

Proximal gradient method

$$x^{k+1} = \text{prox}_{\lambda h}\left(x^k - \lambda^k \nabla f(x^k)\right) : \text{solves} \left(\min f(x) + h(x)\right)$$

*Some special cases:

$$g = I_C \Rightarrow x^{k+1} = \text{prox}_{\lambda^k g}\left(x^k - \lambda^k \nabla f(x^k)\right)$$

$$= \text{prox}_{\lambda^k I_C}\left(x^k - \lambda^k \nabla f(x^k)\right)$$

$$= \Pi_C\left(x^k - \lambda^k \nabla f(x^k)\right)$$

[prox λ I C (□)= Π C (□)]

just projected gradient method

Interpretations:

Majorization minimization interpretation

Majorization minimization: Large class of algorithms $\supseteq$ {gradient method, Newton's method, Expectation minimization algorithm}

goal: $\min_x \; \varphi(x) : \mathbb{R}^n \rightarrow \mathbb{R}$

majorization minimization.

$$x^{k+1} = \underset{x}{\text{argmin}} \; \hat{\varphi}(x, x^k)$$

def: majorization

$\hat{\varphi}(\square, x^k)$ {majorization of $\varphi(x)$}

$\Updownarrow$

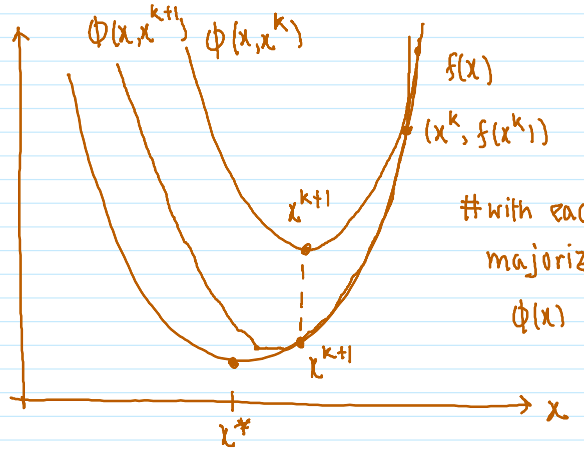
$\hat{\varphi}(\square, x^k)$: (convex upperbound, tight in x^k)

need to check, however I think it is fine

$\forall_{x^k} \forall_x \; \hat{\varphi}(x, x^k) \geq \varphi(x)$, $\varphi(x) = \hat{\varphi}(x, x)$,

$\hat{\varphi}(x, x)$: $\square_x$

- At each step:



with each iteration we are approaching x^*, with majorization $\hat{\varphi}(x, x^k)$ resembling the original function $\varphi(x)$ more and more

% For a convex function at each iteration we move strictly to the optimal point because of the convex nature of the function, that is why majorization-minmization type function always converges for convex optimization problem

1) We majorize (upper bound) the objective

2) Minimize the majorization

% At each iteration in $\hat{\varphi}(x, x^k)$, x^k is changing, as a result the majorization (the upper bound funciton) will also change in shape

an upper bound of f,

$$\hat{f}_\lambda(x,y) = f(y) + \nabla f(y)^T(x-y) + \frac{1}{2\lambda}\|x-y\|_2^2$$ # [lemma: Underestimation Lemma]

this is a majorization as

(i) $x = y \rightarrow \hat{f}_\lambda(x,x) = f(x)$

(ii) $\forall_{\lambda \in (0, \frac{1}{L}]} \ \forall_x \ \forall_y \ f(x) \leq \hat{f}_\lambda(x,y)$

$\therefore x^{k+1} = \underset{x}{\operatorname{argmin}} \ \hat{f}_\lambda(x, x^k)$ will be a majorization-minimization algorithm

অনুযায়ী:

$f : D$, continuously differentiable, Lipschitz continuous with L constant $\Rightarrow \forall_{x,y} \ 0 \leq f(x) - f(y) - \nabla f_0(x)^T(x-y) \leq \frac{L}{2}\|x-y\|_2^2$

$$\rightarrow f(x) \leq f(y) + \nabla f(x)^T(x-y) + \frac{L}{2}\|x-y\|_2^2 \leq f(y) + \nabla f(x)^T(x-y) + \frac{L_+}{2}\|x-y\|_2^2$$

$L_+ \geq L > 0$

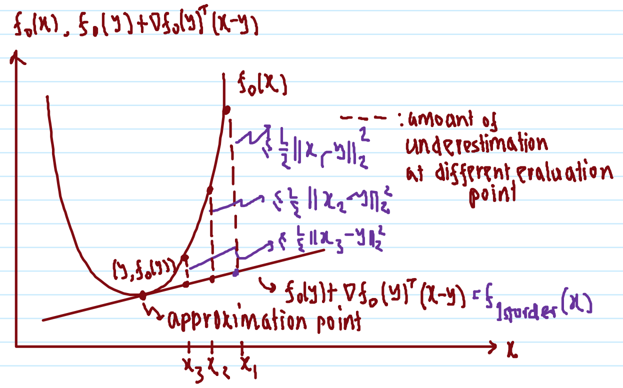
#Intuitive Explanation behind Beck line search condition

now we do not know in L in most problems but by tuning a paramter $\lambda > 0$ if we can arrive at $f(x) \leq f(y) + \nabla f_0(x)^T(x-y) + \frac{1}{2\lambda}\|x-y\|_2^2$ then $\frac{1}{\lambda} = L_+ \geq L$

$\rightarrow \lambda \leq \frac{1}{L}$

$0 < \lambda \leq \frac{1}{L}$

$$\therefore \forall_{\lambda \in (0, \frac{1}{L}]} \ f(x) \leq \underbrace{f(y) + \nabla f_0(x)^T(x-y) + \frac{1}{2\lambda}\|x-y\|_2^2}_{\hat{f}_\lambda(x,y)}$$



Now lets show that proximal gradient too is a majorization-minimization algorithm

Proximal gradient algorithm is

$$x^{k+1} = \mathrm{prox}_{\lambda^k g}\left(x^k - \lambda^k \nabla f(x^k)\right)$$ # $\mathrm{prox}_{\lambda f}(z) = \left(x \,//\, f(\square) \,//\, \square + \frac{1}{2\lambda}\|z - x\|_2^2 \,//\, \sup_x \square \right)$

$$= \left(x \,//\, \lambda^k g(\square) \,//\, \square + \frac{1}{2\lambda}\|x^k - \lambda^k \nabla f(x^k) - x\|_2^2 \,//\, \sup_x \square \right)$$

$$= \underset{x}{\operatorname{argmin}} \left(\lambda^k g(x) + \frac{1}{2}\underbrace{\|x - x^k + \lambda^k \nabla f(x^k)\|_2^2}\right)$$

$$\|(x - x^k) + \lambda^k \nabla f(x^k)\|_2^2$$
$$= \|x - x^k\|_2^2 + (\lambda^k)^2\|\nabla f(x^k)\|_2^2 + 2\lambda^k \nabla f(x^k)^T(x - x^k)$$

$$= \underset{x}{\operatorname{argmin}} \left(\lambda^k g(x) + \frac{1}{2}\|x - x^k\|_2^2 + \underbrace{\frac{1}{2}(\lambda^k)^2\|\nabla f(x^k)\|_2^2}_{\text{constant w.r.t } x \text{, so বাদ দিলেও argmin same থাকবে}} + \lambda^k \nabla f(x^k)^T(x - x^k) \right)$$

$$= \underset{x}{\operatorname{argmin}} \left(\lambda^k g(x) + \frac{1}{2}\|x - x^k\|_2^2 + \lambda^k \nabla f(x^k)^T(x - x^k) \right)$$

$$= \underset{x}{\operatorname{argmin}} \left(\lambda^k g(x) + \frac{1}{2}\|x - x^k\|_2^2 + \lambda^k \nabla f(x^k)^T(x - x^k) + \underbrace{\lambda^k f(x^k)}_{\text{constant w.r.t } x \text{, so add করলেও argmin same থাকবে}} \right)$$

$$= \underset{x}{\operatorname{argmin}} \left(\lambda^k g(x) + \lambda^k \left(f(x^k) + \nabla f(x^k)^T(x - x^k) + \frac{1}{2\lambda^k}\|x - x^k\|_2^2 \right) \right)$$

$$= \underset{x}{\operatorname{argmin}} \left(\lambda^k \left(g(x) + \left(f(x^k) + \nabla f(x^k)^T(x - x^k) + \frac{1}{2\lambda^k}\|x - x^k\|_2^2 \right) \right) \right)$$

λ^k constant w.r.t x, so বাদ দিলেও argmin same থাকবে

Note that if \del f is globally lipschitz, then finding one valid \lambda is sufficient, however when f is not so, we have to modify it.
Now suppose \del f is not globally lipschitz, even then we can use the condition above locally. Assume \del f is locally lipschitz now which implies the derivative is continuous in the relevant domain; this is a much more realistic condition. So, no matter what we will have a local Lipschitz constant as we move from on iterate to the next, Beck line search essentially does just that: finding an approximation of that local Lipshcitz constant by tuning \lambda at each iterate. On the assumption that we have not moved too much from the previous iterate, that is why we apply previous \lambda^{k-1} to see if we got lucky and that \lambda still produces a valid upper bound, if not the only possibility is that we have to make \lambda smaller thus 1/2 \lambda bigger as we move to another point. From a majorization minimization point of view we are tuning \lambda in such a manner that is making \hat{f}_{\lambda}(x,x^k) a majorization

$$= \underset{x}{\operatorname{argmin}} \left(g(x) + \underbrace{f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{1}{2\lambda^k} \|x - x^k\|_2^2}_{\hat{f}_\lambda (x, x^k)} \right)$$

$\hat{f}_\lambda (x, y) = f(y) + \nabla f(y)^T (x-y) + \frac{1}{2\lambda} \|x-y\|_2^2$

$$= \underset{x}{\operatorname{argmin}} \left(g(x) + \hat{f}_\lambda (x, x^k) \right) = \underset{x}{\operatorname{argmin}} \left(q_\lambda (x, x^k) \right)$$

Lets define $g(x) + \hat{f}_\lambda (x,y) = q_\lambda (x,y)$ clearly this is a majorization of $g(x)+f(x)$ $\forall_y \forall_x \forall_{\lambda \in (0, \frac{1}{L}]}$

$\forall_{\lambda \in (0, \frac{1}{L}]} \forall_x \forall_y \quad f(x) \le \hat{f}_\lambda (x,y) \wedge f(x) = \hat{f}_\lambda (x,x)$

$\forall_{\lambda \in (0, \frac{1}{L}]} \forall_y \forall_y \quad \underbrace{g(x) + f(x) \le g(x) + \hat{f}_\lambda (x,y)}_{= q_\lambda (x,y)} \wedge \underbrace{g(x) + f(x) = g(x) + \hat{f}_\lambda (x,x)}_{= q_\lambda (x,x)}$ # same term add

$\therefore \quad x^{k+1} = \operatorname{prox}_{\lambda^k g} \left(x^k - \lambda^k \nabla f(x^k) \right) = \underset{x}{\operatorname{argmin}} \underbrace{q_{\lambda^k} (x, x^k)}_{\text{majorization of } g(x)+f(x)}$

Where to get a valid majorization function at each update
\lambda^{k}=Beck-Teboulle proximal gradient updater(x^k, \lambda^{k-1}, \beta \in (0,1)) if L is not known, or
if L is known \lambda \in (0, 1/L] will work and

$\therefore$ So, proximal algorithm is a majorization minimization algorithm

[Forward-backward splitting]

* Fixed point iteration; # This also works as a proof that proximal algorithm indeed finds the optimal solution of minimize f(x)+g(x)

$$x^* = \underset{x}{\operatorname{argmin}} f(x) + g(x) \Leftrightarrow \partial(f(x) + g(x))_{x=x^*} \ni 0 \Leftrightarrow \nabla f(x^*) + \partial g(x^*) \ni 0$$

$\partial f(x) + \partial g(x)$
$= \nabla f(x) + \partial g(x)$ # f: differentiable

$$\Leftrightarrow \forall_{\lambda > 0} \quad \lambda \nabla f(x^*) + \lambda \partial g(x^*) \ni 0 \quad \Leftrightarrow \quad -\underbrace{(I - \lambda \nabla f)(x^*)}_{\text{vector}} + \underbrace{(I + \lambda \partial g)(x^*)}_{\text{set}} \ni 0$$

$\lambda \nabla f(x^*) - x^* + x^* + \lambda \partial g(x^*)$
$-\underbrace{(I - \lambda \nabla f)(x^*)}_{\text{vector}} \quad \underbrace{(I + \lambda \partial g)(x^*)}_{\text{set}}$

$$\Leftrightarrow \exists_{\eta \in (I + \lambda \partial g)(x^*)} \quad -(I - \lambda \nabla f)(x^*) + \eta = 0$$

$$\Leftrightarrow \exists_{\eta \in (I + \lambda \partial g)(x^*)} \quad \eta = (I - \lambda \nabla f)(x^*)$$

$$\Leftrightarrow (I + \lambda \partial g)(x^*) \ni (I - \lambda \nabla f)(x^*) = z$$

$R(x) \ni y \Leftrightarrow x \in R^{-1}(y)$

$\because (R(x) \ni y) // R^{-1} = R^{-1} R(x) (\ni = \in) R^{-1} y$
$= x \in R^{-1} y$

$$\Downarrow \quad \text{*\% } (R(x) \ni y) // R^{-1} = R^{-1}R(x)(\ni = \in) R^{-1}y = x \in R^{-1}y$$

$$x^* \in (I+\lambda\partial g)^{-1}(I-\lambda\nabla f)(x^*) = \Big(x^* \,//\, (I\square-\lambda\nabla f(\square))(\square) : \text{a vector} \,//\, \overbrace{(I\square+\lambda\partial g(\square))^{-1}}^{\text{function}}\square = \text{singleton}\Big)$$

but we know that resolvent of subdifferential relation is one-to-one, i.e., its a function, so x^* is the only element of $(I+\lambda\partial g)^{-1}(I-\lambda\nabla f)(x^*)$

thm: proximal operator is the resolvent of subdifferential operator

also note $\mathrm{prox}_{\lambda g}(x) = (I+\lambda\partial g)^{-1}(x)$

$$\Leftrightarrow \boxed{x^* = (I+\lambda\partial g)^{-1}(I-\lambda\nabla f)(x^*)}$$

This is the key theorem says that the optimal solution of
$f(x)+g(x)$ can be found by solving the fixed point equation: $\square = (I\square+\lambda\partial g(\square))^{-1}(I\square-\lambda\nabla f(\square))\square$

$$= (I+\lambda\partial g)^{-1}(x^* - \lambda\nabla f(x^*))$$ # $(I+\lambda\partial g)^{-1}(x^*) = \mathrm{prox}_{\lambda g}(x^*)$

$$= \mathrm{prox}_{\lambda g}(x^* - \lambda\nabla f(x^*))$$

$$x^* = \mathrm{argmin}\,(\underbrace{f(x)}_{\text{differentiable}} + \underbrace{g(x)}_{\text{nondifferentiable}}) \Leftrightarrow x^* = \mathrm{prox}_{\lambda g}(x^* - \lambda\nabla f(x^*)) = \underbrace{(I+\lambda\partial g)^{-1}(I-\lambda\nabla f)}_{\text{Forward-backward operator}}(x^*)$$

[Forward-Backward Version of Proximal Gradient Method] [Forward-backward splitting]

$\therefore$ x^* minimizes $f+g$ $\Leftrightarrow$

x^* = fixedpoint (Forward-backward operator)

$\lambda \in (0, \frac{1}{L}]$ ⇒ forward-backward operator is averaged ⇒ iteration converges to a fixed point.

(L: a Lipschitz constant for ∇f)

Arriving at the proximal gradient algorithm from this theorem x^*=(I+\lambda \partial{g})^{-1} (I+\lambda \nabla f)(x*)

We know for any monotone operator f, if x^*=f(x^*), then x^* can be found by following iteration:
x^{k+1}=f(x^{k})

x^*=(I+\lambda \partial{g})^{-1} (I+\lambda \nabla f) (x*) % Assuming the forward-backward operator is monotone then the following iteration will converge to x^*

x^{k+1}=(I+\lambda \partial{g})^{-1} (I+\lambda \nabla f) (x^{k})

=\prox_{\lambda g}{(x^{k}+\lambda \nabla f(x^{k}))}

4.3. Accelerated proximal gradient method:

* Accelerated Proximal Gradient Method:

In a proximal gradient algorithm we do the following:

current iterate

* Accelerated Proximal Gradient Method:

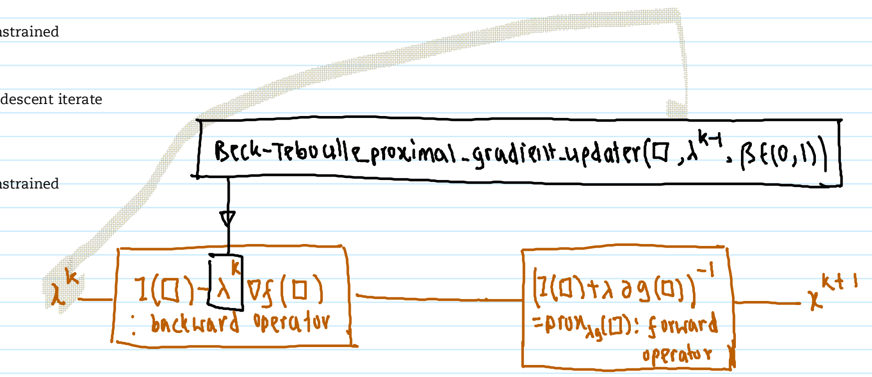
In a proximal gradient algorithm we do the following:

current iterate
// find the gradient descent schemed next iterate for the smooth function as if the non-smooth function does not exist and we are doing a unconstrained optimization of that smooth function using gradient descent
// now we proximize that gradient descent iterate over the nonsmooth function scaled by the step size = next iterate

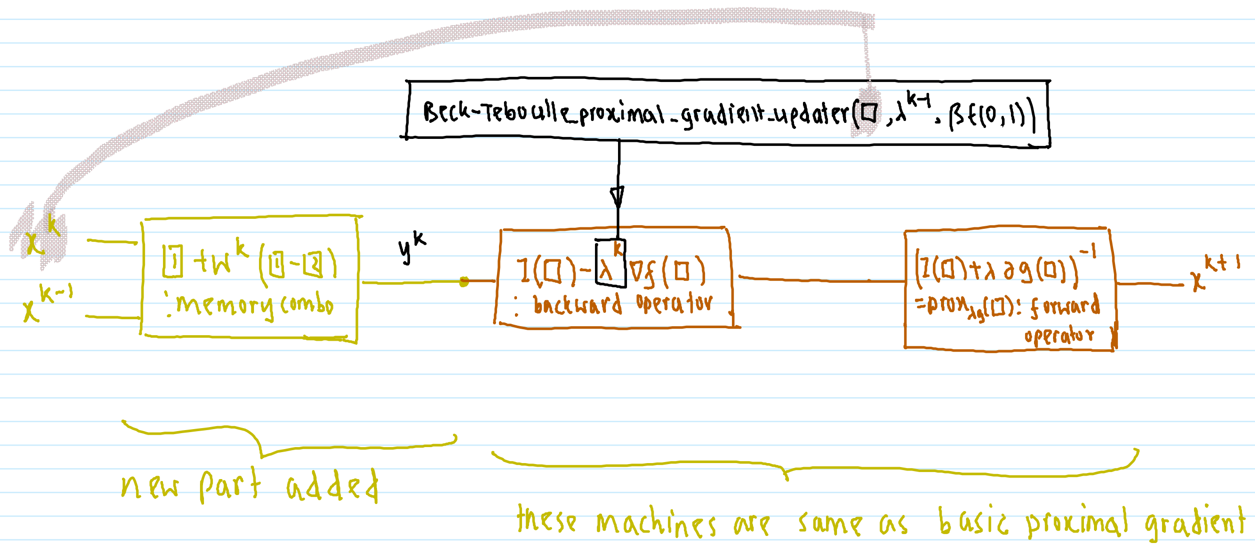
Accelerated proximal gradient algorithm is is similar to proximal gradient step, except it needs more memory and takes proximizes the gradient descent iterate of an intermediate point (which is a linear combination of two previous iterates)

current iterate and the previous iterate
// take a specific linear combination of them = pseudo current iterate
// find the gradient descent schemed next iterate for the smooth function as if the non-smooth function does not exist and we are doing a unconstrained optimization of that smooth function using gradient descent
// now we proximize that gradient descent iterate over the nonsmooth function scaled by the step size = next iterate

in Block diagram: (normal) proximal gradient method:



in accelerated proximal gradient method:



Mathematically:

Accelerated proximal gradient update scheme works as follows:

$$\underbrace{(x^k}_{\square[1]}, \underbrace{x^{k-1})}_{\square[2]} \;//\; \underbrace{\square[1] + w^k(\square[1] - \square[2]) = y^{k+1}}_{\text{intermediate iterate}} : w^k = \frac{k}{k+3} \;//\; \underbrace{\square - \lambda^k \nabla f(\square)}_{\substack{\text{gradient descent} \\ \text{scheme applied to the} \\ \text{intermediate point}}} : \lambda^k: \text{Beck-Teboulle\_proximal\_gradient\_updater}(y^k, \lambda^{k-1}, \beta \in (0,1)) \;//\; \operatorname{prox}_{\lambda^k g}(\square)\Big)$$

Convergence:

$\Big(\nabla f$: Lipschitz continuous, Lipschitz constant $L\Big)$, $\lambda^k \in (0, \frac{1}{L}]$ v λ^k = Beck-Teboulle_proximal_gradient_updater$(y^k, \lambda^{k-1}, \beta \in (0,1))$

$\Rightarrow$ converges with $O\left(\frac{1}{k^2}\right)$

4.4. Alternating direction method of multipliers

ADMM from Proximal Algorithm

$\gamma\ f(x)+g(x)$ # $\{f,g\}\in: \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, $[\underbrace{D}_{proper}]$.

$= \gamma\ f(x)+g(z)$ ⅂ $x-z=0$

[Closed proper convex function] ≝ epi f [⅂∅, [ ], D]

• **Alternating Direction Method of Multipliers:**

(Douglas-Rachford splitting)

eq: ADMM from prox alg

$x^{k+1} = \text{prox}_{\lambda f}(z^k - u^k)$

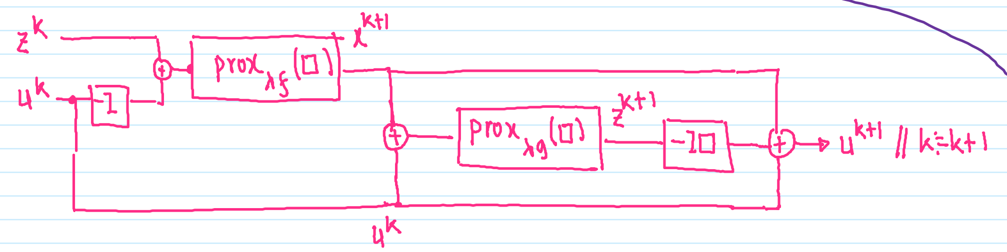
$z^{k+1} = \text{prox}_{\lambda g}(x^{k+1} + u^k)$

$u^{k+1} = u^k + x^{k+1} - z^{k+1}$

Anthropomorphized:
ADMM has 3 iterates
u_k is the running sum of the errors in the 1st iterate and the second iterate

both the 1st and 2nd iterates should become same as iterations progress i.e., they become the optimal solution

In an algorithmic flow:



अर्थात्, (z^k, u^k) // $\square - \square$ // $\text{prox}_{\lambda f}(\square) = x^{k+1}$ // $\square + u^k$ // $\text{prox}_{\lambda g}(\square) = z^{k+1}$ // $-\square$ // $u^k + x^{k+1} + \square = u^{k+1}$ // $k := k+1$

% x is variable of f() so $\text{prox}_{\lambda f}(\cdot) = x^{k+1}$, similarly z is the variable of g(), so $\text{prox}_{\lambda g}(\cdot) = z^{k+1}$

Remember the underlying optimization problem:

$$\begin{pmatrix} \gamma\ f(x)+g(z) \\ ⅂\quad x-z=0 \end{pmatrix}$$

Kothay: z^k is the variable of the nonsmooth function
x^k is the variable of the smooth function
as they are essentially the same variable connected with the equality constraint x=z, the difference in x^k and z^k will converge to zero. Now u^k is the running sum of the difference.

in ADMM $x_k \to x^*$, $z_k \to x^*$, $x_k \to z_k$

Difference between x^k and z^k:

$x^k = \{f\} \to x^k \in \text{dom} f$

$z^k = \{g\} \to z^k \in \text{dom} g$

↳ satisfy the constraints as g encodes constraints

↳ x^k satisfies constraints only in the limit

• **Advantages of ADMM:**

* the objective terms are handled seperately
(functions are accessed only through their proximal operators)

Previously in proximal gradient method we handled both of the functions in one forward backward operator action

* most useful proximal operators of f, g easy to proximal map
but f+g is not easy to evaluate.

• ADMM when $f = I_C(x)$, $g = I_D(x)$

$(\gamma\ f(x)+g(x)) ≝ \text{find } x \in C \cap D$

// $\text{prox}_{I_X}(x) = [x]_X = \Pi_C(x)$

then the ADMM algorithm becomes:

$x^{k+1} = \Pi_C(z^k - u^k)$

$z^{k+1} = \Pi_D(x^{k+1} + u^k)$

note that the parameter λ doesnot appear

$x^{k+1} = \Pi_C(z^k - u^k)$
$z^{k+1} = \Pi_D(x^{k+1} + u^k)$
$u^{k+1} = u^k + x^{k+1} - z^{k+1}$

} note that the parameter λ doesnot appear

↳ in practice the iteration is much faster.

* Interpretation of ADMM: 1) Integral control of a dynamical system:

$x^{k+1} = \operatorname{prox}_{\lambda f}(z^k - u^k)$

$z^{k+1} = \operatorname{prox}_{\lambda g}(x^{k+1} + u^k)$

$u^{k+1} = u^k + x^{k+1} - z^{k+1}$

state z

control u

output to be tracked: x which we want to drive to z the state

$\Rightarrow e^{k+1} = x^{k+1} - u^{k+1}$ is the error signal

$u^{k+1} = u^k + e^{k+1} = u^{k-1} + e^k + e^{k+1} = u^{k-2} + e^{k+1} + e^k + e^{k-1} = u^1 + (e^{k+1} + e^k + e^{k-1} + \dots + e^2)$

$u^k = u^{k-1} + e^k$

$u^{k-1} = u^{k-2} + e^{k-1}$

$^2 = u^1 + e^2$

$= u^1 + \underbrace{\sum_{i=2}^{k+1} e^i}_{\text{running sum of the error signals}}$

2) Augmented Lagrangians:

$(\min f(x) + g(x))$

$\Leftrightarrow \begin{pmatrix} \min & f(x) + g(z) \\ \text{s.t.} & x - z = 0 \end{pmatrix}$ # consensus form

analogous to $u(t) = u(1) + \int_1^t e(\tau)\, d\tau$ = {integral control method}

where the error signal is driven to zero by feeding back the integral of error to its input.

{Augmented Lagrangian} $= L_\rho(x, z, y) = f(x) + g(z) + y^T(x - z) + \frac{\rho}{2}\underbrace{\|x - z\|_2^2}_{\text{additional quadratic penalty term}}$

dual variable (y)

primal variable (1) (x)

primal variable (2) (z)

Classical form of ADMM:

$x^{k+1} := \operatorname{argmin}_x L_\rho(x, z^k, y^k)$ // L_ρ is minimized w.r.t x over the most recent values of the other primal variable (z^k) and dual variable (y^k)

$z^{k+1} := \operatorname{argmin}_z L_\rho(x^{k+1}, z, y^k)$ // " " " " " " " " " " primal variable (x^{k+1}) and dual variable (y^k)

determined in the

$z^{k+1} := \underset{z}{\operatorname{argmin}}\ L_\rho(x^{k+1}, z, y^k)$ // " " " " " " " " " " " primal variable (x^{k+1}) and dual variable (y^k) # determined in the # previous step

$y^{k+1} := y^k + \rho(x^{k+1} - z^{k+1})$ // running sum of the consensus error

// $y^{k+1} = y^k + \rho \sum_{i=2}^{k+1} e^i$, because:

// $y^{k+1} = y^k + \rho\underbrace{(x^{k+1} - z^{k+1})}_{e^{k+1}} = y^k + \rho e^{k+1} = y^{k-1} + \rho(e^{k+1} + e^k) = y^1 + \rho \sum_{i=2}^{k+1} e^i$

// $y^k = y^{k-1} + \rho e^k$

⋮

// $y^2 = y^1 + \rho e^2$

• How the classical version of ADMM leads to the proximal version of ADMM?

[illegible] as const w.r.t z^k

$x^{k+1} = \underset{x}{\operatorname{argmin}}\ L_\rho(x, z^k, y^k) = \underset{x}{\operatorname{argmin}} \left(f(x) + g(z^k) + (y^k)^T(x - z^k) + \frac{\rho}{2}\|x - z^k\|_2^2 \right)$

$f(x) + (y^k)^T x - (y^k)^T z^k + \frac{\rho}{2}\underbrace{\|x - z^k\|_2^2}_{(x-z^k)^T(x-z^k) = x^Tx + (z^k)^T z^k - 2x^T z^k} = f(x) + (y^k)^T x - (y^k)^T z^k + \frac{\rho}{2}\left(x^T x + (z^k)^T z^k - 2x^T z^k\right)$

now we are goal is to write the entire term as $\|x + \boxed{\cdot}\|_2^2 + \boxed{\cdot}$ ← some constant form

Calculation trick: no one will admit: (treat the vectors as numbers first, write them as $(x + \boxed{\cdot})^2 + \boxed{\cdot}$ and then backculculate as vectors:

$\boxed{1}$ Treat the vectors as numbers and find $(x + \boxed{\cdot})^2 + \boxed{\cdot}^2$

$g(x) = yx - yz + \frac{\rho}{2}(x^2 + z^2 - 2xz)$

$= \frac{\rho}{2}\left(x^2 + \overbrace{z^2}^{\text{constnt}} - 2xz + \frac{2}{\rho}yx - \overbrace{\frac{2}{\rho}yz}^{\text{cnstn}}\right)$

$= \frac{\rho}{2}\left(x^2 + 2\left(\frac{1}{\rho}y - z\right)x + \left(\frac{1}{\rho}y - z\right)^2 - \left(\frac{1}{\rho}y - z\right)^2 + z^2 - \frac{2}{\rho}yz\right)$

$= \frac{\rho}{2}\left(\left(x + \frac{1}{\rho}y - z\right)^2 + \left(\underbrace{-\left(\frac{1}{\rho}y - z\right)^2 + z^2 - \frac{2}{\rho}yz}_{\text{constant}}\right)\right)$

part 2 vector extension:

$(y^k)^T x - (y^k)^T z + \frac{\rho}{2}\left(\|x - z^k\|_2^2\right) = \frac{\rho}{2}\left(\|x + \frac{1}{\rho}y^k - z^k\|_2^2\right) + \text{constant}$

$= \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{\rho}{2}\|x - z^k + \frac{1}{\rho}y^k\|_2^2 + \text{constant w.r.t } x \right)$

can be dropped and argmin will remain the same

$= \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{\rho}{2}\|x - z^k + \frac{1}{\rho}y^k\|_2^2 \right)$

$\leftrightarrow\ x^{k+1} = \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{\rho}{2}\|x - z^k + \frac{1}{\rho}y^k\|_2^2 \right)$

lets take, $u^k = \frac{1}{\rho}y^k$, and $\lambda = \frac{1}{\rho}$

$= \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{1}{2}\cdot\frac{1}{\lambda}\|x - z^k + u^k\|_2^2 \right)$

$\operatorname{prox}_{\lambda f}(z^k - u^k) = \underset{fa}{\operatorname{argmin}} \left(\lambda f(fa) + \frac{1}{2}\|fa - (z^k - u^k)\|_2^2 \right) = \underset{fa}{\operatorname{argmin}}\ \lambda\left(f(fa) + \frac{1}{2\lambda}\|fa - z^k + u^k\|_2^2 \right) = \underset{fa}{\operatorname{argmin}} \left(f(fa) + \frac{1}{2\lambda}\|fa - z^k + u^k\|_2^2 \right)$

because constant term common [illegible]

$\operatorname{prox}_h(x) = \operatorname{argmin}\left(h(z) + \frac{1}{2}\|z - x\|_2^2 \right)$ #

$\mathrm{prox}_h(x) = \underset{z}{\mathrm{argmin}}\left(h(z) + \frac{1}{2}\|z - x\|_2^2\right)$]#

$\downarrow$ because constant term common하니 minimizer change 안됨. so λ can be dropped

$= \mathrm{prox}_{\lambda f}(z^k - u^k)$

$\therefore\ x^{k+1} = \mathrm{prox}_{\lambda f}(z^k - u^k)$ [eq: x^{k+1} ADMM classic to proximal]

similarly,

$z^{k+1} = \underset{z}{\mathrm{argmin}}\ L_\rho(x^{k+1}, z, y^k) = \underset{z}{\mathrm{argmin}}\left(f(x^{k+1}) + g(z) + \underbrace{(y^k)^T(x^{k+1} - z)}_{(y^k)^T x^{k+1} - (y^k)^T z} + \frac{\rho}{2}\|x^{k+1} - z\|_2^2\right)$

$f(x^{k+1})$ → # 상수 (대체 가능)

$(y^k)^T x^{k+1}$ → # 상수 (대체 가능)

$= \underset{z}{\mathrm{argmin}}\left(g(z) - (y^k)^T z + \frac{\rho}{2}\|x^{k+1} - z\|_2^2\right) = \underset{z}{\mathrm{argmin}}\left(g(z) + \frac{\rho}{2}\|x^{k+1} - z + \frac{1}{\rho}y^k\|_2^2 + \text{constant}\right) = \underset{z}{\mathrm{argmin}}\left(g(z) + \frac{\rho}{2}\|z - x^{k+1} - u^k\|_2^2\right)$ #[$u^k = \frac{1}{\rho}y^k$, $\lambda = \frac{1}{\rho}$]# $= \underset{z}{\mathrm{argmin}}\left(g(z) + \frac{1}{2\lambda}\|z - x^{k+1} - u^k\|_2^2\right)$

constant → can drop

$\|x^{k+1} - z\|_2^2$: $(x^{k+1} - z)^T(x^{k+1} - z) = \|x^{k+1}\|_2^2 + \|z\|_2^2 - 2z^T x^{k+1}$

#

$-(y^k)^T z + \frac{\rho}{2}\left(\|x^{k+1}\|_2^2 + \|z\|_2^2 - 2z^T x^{k+1}\right)$

$= \frac{\rho}{2}\left(\|x^{k+1}\|_2^2 + \|z\|_2^2 - 2z^T x^{k+1} - \frac{2}{\rho}(y^k)^T z\right)$

#[scalarization 후, counter droping

$\Downarrow$

$x^2 + z^2 - 2zx - \frac{2}{\rho}yz = z^2 - 2\left(x + \frac{1}{\rho}y\right)z + \left(x + \frac{1}{\rho}y\right)^2 \underbrace{- \left(x + \frac{1}{\rho}y\right)^2 + x^2}_{\text{constant}} = \left(z - x - \frac{1}{\rho}y\right)^2 + \text{constant}$

$\Downarrow$ vectorization 후, counter adding

$\|z - x^{k+1} - \frac{1}{\rho}y^k\|_2^2 + \text{constant}$]#

$= \frac{\rho}{2}\left(\|z - x^{k+1} - \frac{1}{\rho}y^k\|_2^2 + \text{constant}\right)$

$= \frac{\rho}{2}\left(\|x^{k+1} - z + \frac{1}{\rho}y^k\|_2^2 + \text{constant}\right)$ return =

$= \underset{z}{\mathrm{argmin}}\ \frac{1}{\lambda}\left(\lambda g(z) + \frac{1}{2}\|z - (x^{k+1} + u^k)\|_2^2\right) = \underset{z}{\mathrm{argmin}}\left(\lambda g(z) + \frac{1}{2}\|z - (x^{k+1} + u^k)\|_2^2\right)$

$\frac{1}{\lambda}$ → can drop it as argmin will stay the same

$= \mathrm{prox}_{\lambda g}(x^{k+1} + u^k)$ #[$\mathrm{prox}_h(x) = \underset{z}{\mathrm{argmin}}\left(h(z) + \frac{1}{2}\|z - x\|_2^2\right)$]#

$= \frac{\rho}{2}\left(\| x^{k+1} - z + \frac{1}{\rho} y^k \|_2^2 + \text{constant} \right)$

$\therefore z^{k+1} = \mathrm{prox}_{\lambda g}(x^{k+1} + u^k)$ [eq: z^{k+1} ADMM classic to proximal]

finally,

$$y^{k+1} := y^k + \rho(x^{k+1} - z^{k+1})$$

$$\Leftrightarrow \underbrace{\frac{1}{\rho} y^{k+1}}_{u^{k+1}} := \underbrace{\frac{1}{\rho} y^k}_{u^k} + (x^{k+1} - z^{k+1})$$

$$\Leftrightarrow u^{k+1} = u^k + (x^{k+1} - z^{k+1})$$

[eq: y^{k+1} =>> u^{k+1} ADMM classic to proximal]

[eq: x^{k+1} ADMM classic to proximal], [eq: z^{k+1} ADMM classic to proximal [eq: y^{k+1} =>> u } [illegible] ADMM classic form to proximal form both are equivalent. :)

Interpretation 3: (Fixed point iteration)

Kind of shows the proof that ADMM works:
We are going to show that if $x^k \sim x^*, z^k \sim z^*, u^k \sim u^*$ then the x^* term will be the minimizer of f(x)+g(x)

$\min_x (f(x) + g(x))$ @ x^* is optimal $\Leftrightarrow 0 \in \partial f(x^*) + \partial g(x^*)$

now

$$\boxed{\begin{aligned} x^{k+1} &= \mathrm{prox}_{\lambda f}(z^k - u^k) \\ z^{k+1} &= \mathrm{prox}_{\lambda g}(x^{k+1} + u^k) \\ u^{k+1} &= u^k + x^{k+1} - z^{k+1} \end{aligned}} \xrightarrow[\#]{x = x^*} \begin{aligned} x^* &= \mathrm{prox}_{\lambda f}(z^* - u^*) \\ z^* &= \mathrm{prox}_{\lambda g}(x^* + u^*) \\ u^* &= u^* + x^* - z^* \Leftrightarrow x^* = z^* \end{aligned}$$

$$\Leftrightarrow \begin{aligned} x^* &= \mathrm{prox}_{\lambda f}(x^* - u^*) = (I + \lambda \partial f)^{-1}(x^* - u^*) \\ x^* &= \mathrm{prox}_{\lambda g}(x^* + u^*) = (I + \lambda \partial g)^{-1}(x^* + u^*) \end{aligned}$$

$x^* = z^*$

Here $(I + □ \partial □)^{-1}$ is the resolvent operator of the proximal mapping of □ □( ) at some point, i.e., $\mathrm{prox}_{□□}(□) = (I + □\partial□)^{-1}(□)$
(□ → sclar, □ → function)
see notes on [1.2 [Boyd] Interpretation of proximal mapping]

$$\Leftrightarrow \begin{aligned} x^* - u^* &\in (I + \lambda \partial f) x^* = x^* + \lambda \partial f(x^*) \\ x^* + u^* &\in (I + \lambda \partial g) x^* = x^* + \lambda \partial f(x^*) \end{aligned}$$

remmember $(I + \partial \lambda f)$ is a relation but
$(I + \partial \lambda f)^{-1}$ is a function

$$\Leftrightarrow \begin{aligned} x^* - u^* &\in x^* + \lambda\{□_i\} = \{x^* + \lambda □_i\} \\ x^* + u^* &\in x^* + \lambda\{□_i\} = \{x^* + \lambda □_i\} \end{aligned}$$

$$\begin{bmatrix} x^* \\ z^* \\ u^* \end{bmatrix} = \begin{bmatrix} \mathrm{prox}_{\lambda f}(□ - □) \\ \mathrm{prox}_{\lambda g}(□ + □) \\ □ + □ - □ \end{bmatrix} \begin{bmatrix} x^* \\ z^* \\ u^* \end{bmatrix}$$

$$\lfloor u' \rfloor \quad \lfloor \boxminus + \boxminus - \boxminus \rfloor \lfloor u^* \rfloor$$

$$x^* + u^* \in x^* + \lambda \{\boxminus_i\} = \{x^* + \lambda \boxminus_i\}$$

$$\Leftrightarrow \begin{cases} \exists\, \boxminus_\zeta \in \partial f(x^*) & x^* - u^* = x^* + \lambda \boxminus_\zeta \\ \exists\, \boxminus_\xi \in \partial g(x^*) & x^* + u^* = x^* + \lambda \boxminus_\xi \end{cases}$$

$$\overset{(+)}{\Rightarrow} \quad 2x^* = 2x^* + \lambda (\underbrace{\boxminus_\zeta + \boxminus_\xi}_{\in \partial (f(x^*) + g(x^*))})$$

$$\{\because \forall_x\ \partial f(x) + \partial g(x) = \partial (f(x) + g(x))\}$$

$$\Leftrightarrow \quad 0 = \lambda (\boxminus_\zeta + \boxminus_\xi)$$

$$\Leftrightarrow \quad 0 \in \partial (f(x^*) + g(x^*))$$

☺