Metodi di Piano di Taglio

Barbarino Giovanni

June 25, 2015

Contents

1 Preliminari

In questa relazione, tratteremo dei Metodi di Piani di Taglio per la minimizzazione di funzioni convesse su domini compatti e poliedrali. Dapprima diamo una formulazione generale di questi metodi, per poi specializzarci a quelli che si basano sui *Bundle*.

1.1 Algoritmo Generale

Nella loro versione generale, questi algoritmi si possono descrivere e analizzare senza fare riferimento alla funzione da minimizzare o al dominio.

Idealmente, vogliamo trovare un punto particolare in una *Regione Ammissibile* $F \subseteq \mathbb{R}^n$, ma visto che in generale questa è complicata da esprimere, prendiamo uno spazio U^0 che "approssima" F, e che sia facile da descrivere. Generiamo dunque una successione di spazi U^k, con lo scopo di avvicinarci sempre di più al punto cercato in F.
Per fare ciò, facciamo riferimento a tre funzioni particolari

- $H(v) = \{\, x \in \mathbb{R}^n \mid a(v) + b(v)^t x \geq 0 \,\}$ porta punti in semispazi.
É la funzione che dà il nome al metodo. Ad ogni passo, andremo a ritagliare gli U^k intersecandoli con il semispazio dato da H. Inoltre ha la proprietà che $U^{k+1} = U^k \cap H(v_k) \neq \emptyset$, e le funzioni a, b sono continue.
- Δ porta punti in insiemi.
Serve per determinare il vettore v su cui verrà costruito il semispazio $H(v)$. In particolare, se w è un vettore non ottimale per il problema, allora $v \in \Delta(w)$ è tale che $w \notin H(v)$.
- Γ porta insiemi in insiemi.
Sarà la funzione che cerca dentro U_k i punti che più probabilmente si avvicinano alla soluzione del problema. In particolare vale $\Gamma(U) \subseteq U \ \ \forall\, U$

L'algoritmo genera dunque una successione di punti $w_k \in \Gamma(U^k)$, ed è strutturato in modo che se $w_k \in F$, allora questa è la soluzione cercata.

Ponendo $k = 0$, l'algoritmo, che d'ora in poi chiameremo (CPG), sarà

$w_0 \in \Gamma(U^0)$
while $w_k \notin F$ **do**
  $v_k \in \Delta(w_k)$
  $U^{k+1} = U^k \cap H(v_k)$
  $w_{k+1} \in \Gamma(U^{k+1})$
  $k = k + 1$
end while
return w_k

Notiamo che se l'algoritmo termina in finiti passi, allora siamo riusciti a trovare una soluzione ottimale. Se invece va avanti indefinitivamente, riusciamo comunque a fornire un primo risultato di convergenza

Teorema 1. *Nel caso in cui*

(a) L'algoritmo (CPG) genera una sequenza infinita di spazi U^k e punti w_k, v_k

(b) Tutti i w_k e i v_k sono contenuti in un insieme compatto

(c) Se $w \notin F$, allora Δ è chiusa in w. Ossia

$$z_k \to w, \quad y_k \in \Delta(z_k), \quad y_k \to y \implies y = \Delta(w)$$

Allora ogni sottosuccessione dei w_k convergente tende alla soluzione ottimale. Grazie alla compattezza, inoltre, ne esiste almeno una.

Proof. Prendiamo una sottosuccessione convergente dei w_k, e da questa ne estraiamo un'altra in modo che anche i v_k corrispondenti convergano. Per semplicità, denoteremo questa sottosuccessione con lo stesso indice k. Poniamo dunque $w_k \to w$, $v_k \to v$. Sappiamo che $v_k \in \Delta(w_k)$ implica, per chiusura, che $v \in \Delta(w)$. Se w non fosse una soluzione ottimale, allora $w \notin H(v)$, ma $w_k \in \Gamma(U^k) \subseteq U^k \subseteq H(v_{k-1})$, ed in generale $w_k \in H(v_h) \;\; \forall\, h < k$. Dato che $H(v)$ è chiuso, allora $w \in H(v_k) \;\; \forall\, k$, e dato che H è pure continua, $w \in H(v)$, assurdo. Dunque w è una soluzione ottimale. □

Forniamo un po' di osservazioni, che permettono di comprendere meglio la natura delle ipotesi del teorema:

- Notiamo innanzitutto che gli U_k generati sono strettamente contenuti gli uni negli altri, poiché

  $$w_k \in \Gamma(U^k) \subseteq U^k \quad v_k \in \Delta(w_k) \implies w_k \notin H(v_k) \implies w_k \notin U^{k+1}$$

  Inoltre, i w_k sono contenuti definitivamente in ogni U_h, dunque basta che esista un U_k limitato per far sì che i w_k siano contenuti in un compatto.

- La condizione *(c)* è una generalizzazione del "Grafico Chiuso", che nel caso in cui Δ porti punti in punti, è equivalente alla continuità. In questo caso, inoltre, se i w_k sono contenuti in un compatto, lo saranno anche i v_k poiché $v_k = \Delta(w_k)$.

- Notiamo infine che se U^0 è facilmente esprimibile, allora lo saranno anche gli U^k, in quanto sono intersezioni di U^0 con vincoli lineari.

Cerchiamo adesso di specificare gli algoritmi nel nostro caso, ossia con funzioni e domini convessi.

1.2 Piano di Taglio Classico

Data una funzione convessa e continua $f(x)$, ed un dominio compatto e poliedrale $C \subseteq \mathbb{R}^n$, il nostro problema sarà

$$(P) \quad \min_{x \in C} f(x)$$

Dato che il dominio è compatto, e la funzione è convessa, allora esisterà sicuramente un minimo in C, che chiameremo $\bar{x}$.

Grazie alla convessità, sappiamo che $f(x)$ è il limite superiore degli iperpiani tangenti dati dai sottogradienti. L'idea, quindi, è di ritagliare via il sottografico di $f(x)$ tramite iperpiani tangenti, e cercare il minimo dello spazio rimanente. Più nello specifico,

- Dati dei punti $\{x_1, x_2, \ldots, x_m\} \in C$, per ognuno ricaviamo un sottogradiente $s_i \in \partial f(x_i)$, e consideriamo gli iperpiani tangenti

$$f_i(y) = f(x_i) + < s_i, y - x_i >$$

- Tutti questi iperpiani sono contenuti nel sottografico di $f(x)$, dunque anche il loro massimo lo sarà

$$\varphi_m(y) = \max_i f_i(y) \qquad \varphi_m(x) \leq f(x) \ \ \forall x \in C$$

- Preso un minimo x_{m+1} di $\varphi_m(x)$ (ricavabile tramite un problema di Programmazione Lineare), avremo che $\varphi_m(x_{m+1}) \leq f(\bar{x})$, e un buon test di ottimalità sarà

$$f(x_{m+1}) - \varphi_m(x_{m+1}) \leq tol$$

- In caso non passasse il test, si aggiorna φ_m e si ripete

$$s_{m+1} \in \partial f(x_{m+1}) \qquad f_{m+1}(y) = f(x_{m+1}) + < s_{m+1}, y - x_{m+1} >$$

$$\varphi_{m+1}(y) = \max\{\varphi_m(y), f_{m+1}(y)\}$$

Prendendo come dati iniziali $k = 1$, $\varphi_0 \equiv -\infty$, $\delta_1 = \infty$, $x_1 \in C$, $tol > 0$, allora il nostro algoritmo (CP) si presenta come

while $\delta_k > tol$ **do**
  $s_k \in \partial f(x_k)$
  $\varphi_k(y) = \max\{\, \varphi_{k-1}(y), f(x_k) + < s_k, y - x_k > \}$
  $x_{k+1} \in \arg\min \varphi_k(y)$
  $k = k + 1$
  $\delta_k = f(x_k) - \varphi_{k-1}(x_k)$
end while
return x_k

Nelle notazioni dell'algoritmo (CPG), diremo che la regione ammissibile F è il sopragrafico, modificato tramite tol, ossia

$$F = \{\ (z, x) \mid f(x) - z \leq tol,\ \ x \in C\ \}$$

e i punti w_k che generiamo sono le coppie $(\varphi_{k-1}(x_k), x_k)$. L'algoritmo infatti termina se $w_k \in F$, ossia se

$$\delta_k = f(x_k) - \varphi_{k-1}(x_k) \leq tol$$

Vediamo ora che riusciamo a ricondurre l'algoritmo (CP) al (CPG), e scegliendo opportunamente le funzioni e gli spazi, soddisfa le ipotesi del Teorema 1

$\boldsymbol{U^k}$ Idealmente, i nostri U^k dovrebbero essere le regioni delimitate dagli iperpiani, ossia $U^k = \{\ (z, x) \mid z \geq \varphi_k(x),\ \ x \in C\ \}$, ma questi non sono compatti, poiché illimitati nella prima componente. Per ovviare a ciò, si prende un limite superiore dato dal punto iniziale x_1:

$$U^k = \{\ (z, x) \mid f(x_1) \geq z \geq \varphi_k(x),\ \ x \in C\ \}$$

Dunque i vettori w_k sono contenuti in un compatto.

$\boldsymbol{H}$ Le funzioni continue che definiscono H saranno

$$a((x, s)) =< s, x > -f(x) \qquad b((x, s)) = (1, -s)$$

$$H((x, s)) = \{\ (z, y) \mid z \geq f(x)+ < s, y - x >\ \}$$

Ovviamente, $U^{k+1} = H(v_k) \cap U^k$ non sarà mai vuoto, in quanto conterrà sempre la coppia di valori ottimali $(f(\bar{x}), \bar{x})$

Δ La funzione Δ ci fornisce i sottogradienti

$$\Delta((z, x)) = \{x\} \times \partial f(x)$$

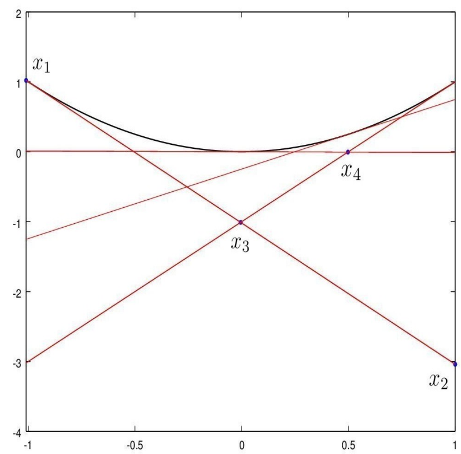
Questa è una funzione chiusa grazie alla continuità e convessità di $f(x)$ sui compatti, ed è fatta in modo che se $(z, x) \notin F$, allora qualsiasi elemento $(x, s) \in \Delta((z, x))$ produca tramite H un iperpiano che taglia fuori (z, x). Le immagini sono contenute in $C \times \cup_{x \in C} \partial f(x)$, che è limitato, e dunque anche i v_k sono contenuti in un compatto.

Γ La funzione Γ risulterà semplicemente

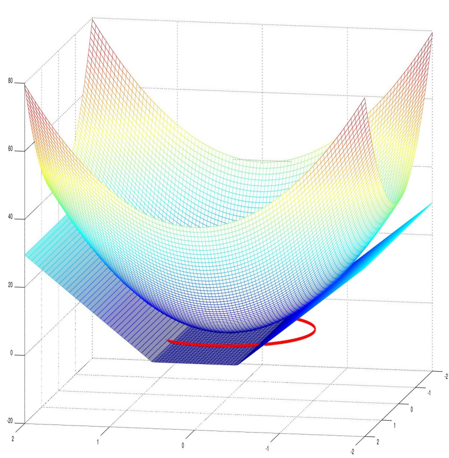
$$\Gamma(V) = \arg\min \{\ z \mid (z, x) \in V\ \}$$

1.3 Problemi

L'algoritmo (CP), nonostante il risultato di convergenza, presenta alcune particolarità che rallentano di molto il metodo.



1. Il metodo non è necessariamente di discesa, ossia non è vero che
$$f(x_{k+1}) \leq f(x_k)$$
2. Quando il metodo arriva molto vicino al minimo, ma non si ferma, l'iperpiano generato all'iterazione successiva è circa orizzontale, e fa allontanare di molto i punti successivi, come si può vedere nella figura a fianco.
3. Quando il numero di funzioni che definiscono φ_k diventa grande, il problema lineare risulta sempre più difficile da risolvere
4. Quando φ_k ha più di un minimo, una scelta sbagliata di quest'ultimo può avere conseguenze disastrose.



Un esempio del quarto problema si riscontra a dimensione $n \geq 2$ (nel disegno $n = 2$), con la funzione $f(x) = \|x\|^2 - 1 + 2\lambda$, dove λ è piccolo. Facendo partire l'algoritmo (CP) dal punto $(0,0)$, al primo passo tutti i punti del dominio sono di minimo per $\varphi_1(y)$. Se imponiamo la regola che quando è possibile scegliamo solo punti di minimo di norma 1, allora prima di avvicinarsi di nuovo a zero, l'algoritmo dovrà eliminare tutti i vettori di norma unitaria con i piani di taglio (nel disegno, il cerchio unitario è disegnato in rosso). É possibile ricavare che ci vogliono almeno

$$k = 2(4\lambda)^{-n/2}$$

passi perché ciò avvenga. Se per esempio $\lambda = 0.025$ e $n = 20$, allora questo numero diventa $k \geq 2 * 10^{10}$, e serve solo per ridurre l'errore di 0.05.

2 Metodi Bundle

I *Metodi Bundle* si presentano in maniera più generale dell'algoritmo (CP). Come anticipato, avremo bisogno di modificare vari step, quali il problema di minimizzazione lineare che ci permetteva di ricavare x_{k+1}, il calcolo dell'errore δ_{k+1}, ed anche la funzione $\varphi_k(y)$.
Nel descrivere questi algoritmi, poniamo che il dominio della nostra funzione $f(x)$ sia tutto $\mathbb{R}^n$ per semplicità, ma è facile riadattare il tutto per un dominio C poliedrale.

Vediamo adesso come si presenta l'algoritmo, che chiamiamo (GB):
Partiamo da $x_1 = y_1 \in C$, $\delta = \infty$, $k = 1$, e con $\varphi_1(y) = f(y_1) + < s_1, y - y_1 >$, dove $s_1 \in \partial f(y_1)$. Fissiamo inoltre due costanti $tol > 0$ e $0 < m < 1$, ed un parametro iniziale $\mu_1 \geq 0$.

while $\delta_k > tol$ **do**
    $y_{k+1} \in \arg\min pbm(\varphi_k, \mu_k, x_k)$
    $\delta_{k+1} = \delta(f, \varphi_k, y_{k+1}, x_k)$
    if $f(x_k) - f(y_{k+1}) \geq m\delta_{k+1}$ **then**
        $x_{k+1} = y_{k+1}$ (descent-step)
    else
        $x_{k+1} = x_k$ (null-step)
    end if
    $\varphi_{k+1} = updmod(f, \mu_k, \varphi_k, y_{k+1}, y_k, x_k)$
    $\mu_{k+1} = updpar(f, \mu_k, y_{k+1}, y_k, x_k)$
    $k = k + 1$
end while
return x_k

Specificheremo le varie funzioni nei prossimi paragrafi. Per ora, invece, parliamo della soluzione al problema 1.

Possiamo notare che l'algoritmo genera due sequenze di punti y_k e x_k. I primi sono la sequenza di punti classica che avevamo anche nel (CP), mentre gli x_k rappresentano una sottosuccessione degli y_k, e sono detti *centri di stabilità*.
La condizione

$$f(x_k) - f(y_{k+1}) \geq m\delta_{k+1} \geq 0$$

fa sì che se due elementi consecutivi x_i non sono uguali, allora $f(x_{k+1}) = f(y_{k+1}) \leq f(x_k)$, assicurando così che gli x_k siano una successione discendente. Il parametro m decide di quanto deve migliorare l'approssimazione ad ogni passo, ossia la "velocità" di convergenza degli x_k.

Un altro comando che potremmo aggiungere al programma, è di fermarsi se trova un sottogradiente di norma abbastanza piccola, anche perché sappiamo che un punto è di minimo se e solo se 0 appartiene al sottogradiente.

Vediamo adesso come risolvere i problemi di oscillazione.

2.1 Penalizzazione

I problemi 2. e 4. erano causati dal fatto che i minimi della funzione φ_k non erano sempre delle buone scelte, a volte perché troppo lontani dai punti precedenti, a volte perché non unici. Una volta che abbiamo dei punti x_k che designiamo come "buoni", possiamo provare a cercare dunque i punti successivi in modo che siano "vicini" agli x_k.

Un metodo per fare questo, è cercare y_{k+1} che risolva il *Problema Penalizzato*

$$y_{k+1} = \arg\min pbm(\varphi_k, \mu_k, x_k) = \arg\min \left(\varphi_k(y) + \frac{1}{2}\mu_k\|y - x_k\|^2\right)$$

La funzione $\varphi_k(y)$ sarà diversa da quella dell'algoritmo (CP), ma in ogni caso rimarrà un massimo di funzioni lineari, dunque *pbm* risulta essere un problema quadratico. Quando $\mu_k > 0$, notiamo inoltre che la funzione da minimizzare diventa fortemente convessa, e pertanto possiede un unico minimo, risolvendo così il problema 4.

Con questa scelta, inoltre, modifichiamo anche l'errore

$$\delta_{k+1} = f(x_k) - \left(\varphi_k(y_{k+1}) + \frac{1}{2}\mu_k\|y_{k+1} - x_k\|^2\right)$$

e, dato che y_{k+1} è il minimo per *pbm*,

$$\delta_{k+1} = f(x_k) - \left(\varphi_k(y_{k+1}) + \frac{1}{2}\mu_k\|y_{k+1} - x_k\|^2\right) \geq f(x_k) - \varphi_k(x_k) \geq 0$$

Notiamo che scegliere un μ_k grande, da un lato non permette agli y_{k+1} di allontanarsi troppo, ma dall'altro rende l'errore piccolo, facendo sì che il programma sia meno rigido nell'accettare i punti come soluzione ottimale. Di solito, si sceglie un parametro μ_1 piccolo, e si aggiorna in modo opportuno passo per passo.

Possiamo ricavare delle osservazioni importanti dal duale del problema penalizzato, che serviranno più avanti. Per fare ciò, abbiamo bisogno di saper descrivere le funzioni lineari che costituiscono $\varphi_k(y)$, che in generale saranno diverse da quelle dell'algoritmo (CP).

Riscriviamo dunque $\varphi_k(y)$ dell'algoritmo classico, rispetto ai punti x_k:

$$\begin{aligned}
\varphi_k(y) &= \max_{i=1,\dots,k} \{\, f(y_i) + < s_i, y - y_i > \} \\
&= f(x_k) + \max_{i=1,\dots,k} \{\underbrace{f(y_i) - f(x_k) + < s_i, x_k - y_i >}_{= -e_i} + < s_i, y - x_k > \} \\
&= f(x_k) + \max_{i=1,\dots,k} \{-e_i + < s_i, y - x_k > \}
\end{aligned}$$

In generale, anche nel nuovo algoritmo (GB), le funzioni lineari contenute in $\varphi_k(y)$ saranno identificate dai valori $(x_k, f(x_k))$, e dalle coppie (e_i, s_i). Queste ultime danno il nome al metodo, in quanto si definisce il *Bundle* al passo k come

$$\mathcal{B}_k = \{ \ (e_i, s_i) \mid i = 1, \dots, n_k \ \}$$

dove n_k è il numero di funzioni che definiscono φ_k al passo k, che in generale saranno meno di k.

Prima di andare avanti, definiamo un nuovo oggetto: chiamiamo *ε-sottodifferenziale* della funzione f nel punto x l'insieme

$$\partial_\varepsilon f(x) = \{ \ s \in \mathbb{R}^n \mid f(y) \geq f(x) + < s, y - x > - \ \varepsilon \ \ \forall y \in \mathbb{R}^n \ \}$$

che essenzialmente descrive tutti gli iperpiani passanti per $(x, f(x) - \ \varepsilon)$ e contenuti nel sottografico di $f(x)$. Nelle nostre ipotesi, ossia di $f(x)$ continua e convessa, l'ε-sottodifferenziale è vuoto se e solo se $\varepsilon < 0$.
Con questa notazione possiamo finalmente enunciare il teorema

Teorema 2. *Dati x_k e y_{k+1} generati dall'algoritmo (GB), allora sono legati da*

$$y_{k+1} = x_k - \frac{1}{\mu_k}\widehat{s_k} \qquad \widehat{s_k} = \sum_{i=1}^{n_k} \bar{\alpha}_i s_i$$

dove il vettore $\bar{\alpha}$ risolve il problema

$$\begin{cases} \min_\alpha \frac{1}{2} \| \sum_{i=1}^{n_k} \alpha_i s_i \| + \mu_k \sum_{i=1}^{n_k} \alpha_i e_i \\ \alpha \in \Delta_k := \{ \ z \in [0,1]^{n_k} \mid \sum_{i=1}^{n_k} z_i = 1 \ \} \end{cases}$$

Inoltre $\widehat{s_k}$ soddisfa

1. $\widehat{s_k} \in \partial \varphi_k(y_{k+1})$
2. $\delta_{k+1} = \varepsilon_k + {\|\widehat{s_k}\|^2}/{2\mu_k}$ *dove* $\varepsilon_k = \sum_{i=1}^{n_k} \bar{\alpha}_i e_i \geq 0$
3. $\widehat{s_k} \in \partial_{\varepsilon_k} f(x_k)$

Proof. Usando

$$\varphi_k(y) = f(x_k) + \max_{i=1,\dots,n_k} \{ -e_i + < s_i, y - x_k > \}$$

riscriviamo il problema penalizzato come

$$\begin{cases} \min_{(z,y)} \left(z + \frac{1}{2}\mu_k \|x_k - y\|^2 \right) \\ z \geq f(x_k) - e_i + < s_i, y - x_k > \quad \forall \ i \end{cases}$$

Usando il teorema di Dualità Forte, sappiamo che non c'è Gap di Dualità, ed esistono gli ottimi duali e primali. Scriviamo dunque la Lagrangiana e il duale

$$L(z, y, \alpha) = z + \frac{1}{2}\mu_k \|x_k - y\|^2 + \sum_{i=1}^{n_k} \alpha_i (f(x_k) - e_i - z + < s_i, y - x_k >)$$

$$\varphi(\alpha) = \inf_{(z,y)} \left[z\left(1 - \sum_{i=1}^{n_k} \alpha_i\right) + \frac{1}{2}\mu_k\|x_k - y\|^2 + \sum_{i=1}^{n_k} \alpha_i(f(x_k) - e_i + < s_i, y - x_k >) \right]$$

Dato che dovremo massimizzare questa funzione, allora possiamo imporre che la somma degli α_i sia uno, altrimenti $\varphi(\alpha) = -\infty$. Dunque

$$\varphi(\alpha) = \inf_{y} \left[\frac{1}{2}\mu_k\|x_k - y\|^2 + f(x_k) - \sum_{i=1}^{n_k} \alpha_i e_i + < \sum_{i=1}^{n_k} \alpha_i s_i, y - x_k > \right] = \inf_{y} \psi(y)$$

La funzione $\psi(y)$ è però differenziabile in y, dunque avrà minimo se

$$\nabla\psi(y) = \mu_k(y - x_k) + \sum_{i=1}^{n_k} \alpha_i s_i = 0 \implies y = x_k - \frac{1}{\mu_k} \sum_{i=1}^{n_k} \alpha_i s_i$$

$$\begin{aligned}
\varphi(\alpha) &= \frac{1}{2\mu_k} \left\| \sum_{i=1}^{n_k} \alpha_i s_i \right\|^2 + f(x_k) - \sum_{i=1}^{n_k} \alpha_i e_i - \frac{1}{\mu_k} \left\| \sum_{i=1}^{n_k} \alpha_i s_i \right\|^2 \\
&= f(x_k) - \frac{1}{\mu_k} \left(\frac{1}{2} \left\| \sum_{i=1}^{n_k} \alpha_i s_i \right\|^2 + \mu_k \sum_{i=1}^{n_k} \alpha_i e_i \right)
\end{aligned}$$

Il problema duale consiste nel massimizzare questa quantità per α positivo, dunque otteniamo esattamente il problema della tesi. Inoltre

$$y_{k+1} = x_k - \frac{1}{\mu_k} \sum_{i=1}^{n_k} \bar{\alpha}_i s_i = x_k - \frac{1}{\mu_k} \widehat{s}_k$$

Dimostriamo ora i risultati su $\widehat{s}_k$:

1. y_{k+1} ottimale implica che 0 appartiene al sottogradiente del problema penalizzato, dunque

$$\begin{aligned}
0 &\in \partial \left(\varphi_k(\cdot) + \frac{1}{2}\mu_k\|x_k - \cdot\|^2 \right)(y_{k+1}) \\
&= \partial\varphi_k(y_{k+1}) + \mu_k(y_{k+1} - x_k) = \partial\varphi_k(y_{k+1}) - \widehat{s}_k
\end{aligned}$$

2. Ponendo il Gap di Dualità uguale a zero, otteniamo proprio che

$$f(x_k) - \delta_{k+1} = \varphi_k(y_{k+1}) + \frac{1}{2}\mu_k\|x_k - y_{k+1}\|^2 = f(x_k) - \varepsilon_k - \|\widehat{s}_k\|^2 / 2\mu_k$$

3. Dai primi due punti, otteniamo

$$\begin{aligned}
\varepsilon_k &= \delta_{k+1} - \|\widehat{s}_k\|^2 / 2\mu_k \\
&= f(x_k) - \left(\varphi_k(y_{k+1}) + \frac{1}{2}\mu_k\|y_{k+1} - x_k\|^2 \right) - \frac{1}{2}\mu_k\|y_{k+1} - x_k\|^2 \\
&= f(x_k) - \varphi_k(y_{k+1}) - \mu_k\|y_{k+1} - x_k\|^2
\end{aligned}$$

$$
\begin{aligned}
&f(x_k) - \varepsilon_k + < \widehat{s_k}, y - x_k > = \\
&= \varphi_k(y_{k+1}) + \mu_k \|y_{k+1} - x_k\|^2 + < \widehat{s_k}, y - x_k > \\
&= \varphi_k(y_{k+1}) - < \widehat{s_k}, y_{k+1} - x_k > + < \widehat{s_k}, y - x_k > \\
&= \varphi_k(y_{k+1}) + < \widehat{s_k}, y - y_{k+1} > \leq \varphi_k(y) \leq f(y)
\end{aligned}
$$

□

Dal punto di vista computazionale, questo teorema ci dice che è possibile calcolare y_{k+1} risolvendo un problema quadratico diverso da quello originale, ma con lo stesso numero di vincoli. I vettori $\bar{\alpha}$ e $\widehat{s_k}$ che ricaviamo, inoltre, portano con sè molte informazioni, e ci permettono, come vediamo nel prossimo paragrafo, di modificare φ_k in modo da renderlo più compatto.

Una nota da fare è che questo ci permette di calcolare δ_{k+1} solo sapendo gli e_i e i s_i, ossia gli elementi del Bundle. Inoltre, dato che $\varepsilon_k \geq 0$, possiamo modificare il metodo di stop come

$$\varepsilon_k < tol_\varepsilon \qquad \text{e} \qquad \|\widehat{s}_k\|^2 < tol_{\widehat{s}}$$

e una scelta opportuna di questi parametri, permettono una maggiore precisione nella soluzione.

2.2 Aggregazione

Cerchiamo qui di venire incontro al Problema 3., ossia la quantità eccessiva di vincoli che compongono φ_k. Il nostro obiettivo è limitare tale numero a un parametro n_{max}, e per fare questo, abbiamo bisogno di un criterio che ci dica quali vincoli possiamo eliminare, e se questo non bastasse, vorremmo in qualche modo comprimere l'informazione contenuta nei vincoli.

Consideriamo $\widehat{s_k}$ del Teorema 2, e definiamo la nuova funzione

$$\widetilde{f_k}(y) = f(x_k) - \varepsilon_k + < \widehat{s_k}, y - x_k >$$

che corrisponderà ad una coppia $(\varepsilon_k, \widehat{s_k})$ da inserire nel bundle, sotto certe condizioni. Dalla dimostrazione dello stesso teorema, sappiamo che

$$\widetilde{f_k}(y) = \varphi_k(y_{k+1}) + < \widehat{s_k}, y - y_{k+1} >$$

ed inoltre

- $\widehat{s_k} \in \partial\varphi_k(y_{k+1}) \implies \widetilde{f_k}(y)$ è tangente a $\varphi_k(y)$ in y_{k+1}
- $\widehat{s_k} \in \partial_{\varepsilon_k} f(x_k) \implies \widetilde{f_k}(y)$ appartiene al sottografico di $f(y)$

Vale anche il seguente risultato:

Lemma 1. *Ogni $\psi(y)$ funzione convessa per cui valgano*

$$1)\ \psi(y_{k+1}) = \widetilde{f}_k(y_{k+1}) \qquad 2)\ \psi(y) \geq \widetilde{f}_k(y)\ \ \forall y$$

soddisfa

$$y_{k+1} = \arg\min \left(\psi(y) + \frac{1}{2}\mu_k \|y - x_k\|^2 \right)$$

Proof. $\widehat{s}_k$ è un sottogradiente di ψ in y_{k+1} poiché

$$\psi(y_{k+1}) + < \widehat{s}_k, y - y_{k+1} >= \varphi_k(y_{k+1}) + < \widehat{s}_k, y - y_{k+1} >= \widetilde{f}_k(y) \leq \psi(y)$$

ma la condizione di ottimalità per y_{k+1} è

$$0 \in \ \partial\psi(y_{k+1}) - \widehat{s}_k$$

□

Questo ci dice che $\widetilde{f}_k$ contiene le informazioni accumulate dall'algoritmo fino a quel momento, ed è sufficiente, da sola, a determinare l'approssimazione y_{k+1} successiva.

Ritornando al Teorema 2, notiamo che $\widehat{s}_k$ è determinato da $\bar{\alpha}$, ma se ci fosse una componente $\bar{\alpha}_i$ nulla, allora l's_i corrispondente non avrebbe ruolo nel determinare $\widehat{s}_k$ e y_{k+1}, e pertanto può essere tolto. Chiamiamo dunque gli iperpiani relativi a tali s_i *vincoli inattivi.*

Possiamo, grazie a ciò, definire la

Compressione Se il numero di vincoli n_k contenuti in φ_k raggiunge n_{max}, allora calcoliamo $\bar{\alpha}$, ed eliminiamo dal bundle i vincoli inattivi.

Può capitare che però non ci sia nessun vincolo inattivo, e in questo caso applichiamo la

Aggregazione Se il numero di vincoli attivi è pari a n_{max}, allora rimuoviamo due vincoli attivi casuali, e introduciamo il vincolo $\widetilde{f}_k$, ossia aggiungiamo al bundle la coppia $(\varepsilon_k, \widehat{s}_k)$

Questi due metodi ci permettono finalmente di definire la funzione *updmod*$(f, \mu_k, \varphi_k, y_{k+1}, y_k, x_k)$ dell'algoritmo (GB)

if $n_k = n_{max}$ **then**
  Applica la Compressione
  Applica l'Aggregazione
end if
calcoliamo $s_{k+1} \in \partial f(y_{k+1})$, e il relativo e_{k+1}
aggiungiamo la coppia (e_{k+1}, s_{k+1}) al bundle
if $x_{k+1} = y_{k+1}$ **then**
  $e_i = e_i - f(x_k) + f(x_{k+1}) + < s_i, x_k - x_{k+1} > \ \ \forall i$
end if

L'ultimo comando viene eseguito quando in (GB) viene fatto un *descent-step*, ossia quando un nuovo centro di stabilità x_{k+1} viene trovato. In questo caso, gli s_i rimangono uguali, ma gli e_i cambiano. Se chiamiamo e'_i i nuovi, avremo

$$\begin{aligned} f_i(y) =& f(x_k) - e_i + < s_i, y - x_k > \\ =& f(x_{k+1}) \underbrace{- e_i + f(x_k) - f(x_{k+1}) - < s_i, x_k - x_{k+1} >}_{= -e'_i} + < s_i, y - x_{k+1} > \end{aligned}$$

Ora che abbiamo anche un metodo per limitare n_k, manca solo un algoritmo per aggiornare i μ_k di passo in passo. Questo si opera tramite la *Reversal Form*, che usa la Regolarizzazione di Moreau-Yoshida e un metodo simil-Eulero per decidere μ_{k+1} imponendo una convergenza al secondo ordine.

I risultati di convergenza sono comunque formulati in relazione ad arbitrarie sequenze μ_k, dunque possiamo enunciarli senza stabilire una regola per aggiornare i μ_k.

2.3 Reversal Form

Consideriamo adesso La *Regolarizzazione di Moreau-Yoshida* per funzioni convesse $f(x)$, ossia, fissato un parametro μ,

$$F_\mu(x) = \min_y \left\{ f(y) + \frac{1}{2}\mu\|y - x\|^2 \right\}$$

e indichiamo con $p_\mu(x)$ il vettore y minimo.
Questo problema ha proprietà interessanti quali

(a) La funzione F_μ è differenziabile, e il suo gradiente è lipschitziano, con

$$\nabla F_\mu(x) = \mu(x - p_\mu(x))$$

(b) Nel punto di minimo $p_\mu(x)$ la funzione deve avere un sottogradiente pari a zero, dunque

$$p_\mu(x) = x - \frac{1}{\mu} s(p_\mu(x)) \qquad s(p_\mu(x)) \in \partial f(p_\mu(x))$$

(c) Se $\mu \geq 0$, allora minimizzare $F_\mu(x)$ equivale a minimizzare $f(x)$.

Ritornando al nostro algoritmo (GB), notiamo che

$$y_{k+1} = \arg\min \left(\varphi_k(y) + \frac{1}{2}\mu_k\|y - x_k\|^2 \right)$$

è molto simile al problema sopra con $\mu = \mu_k$, $x = x_k$, ossia

$$p_{\mu_k}(x_k) = \arg\min \left(f(y) + \frac{1}{2}\mu_k\|y - x_k\|^2 \right)$$

In particolare, se decidiamo di aggiornare μ_k solo quando abbiamo un descend-step, ossia quando x_k cambia, otteniamo che tutti i null-step hanno come obiettivo di avvicinare y_{k+1} a $p_{\mu_k}(x_k)$ il più possibile, in quanto $\varphi_k(y)$ sarà sempre più vicino a $f(x)$. Cerchiamo dunque un modo per determinare il migliore μ_{k+1} in caso di descend-step.

Il nostro obiettivo è minimizzare $f(x)$, o equivalentemente $F_\mu(x)$, ossia trovare una radice di $\nabla F_\mu(x)$. Per fare ciò, utilizziamo un metodo di Eulero, chiamando $L(\mu, x)$ l'hessiana di $F_\mu(x)$ (o una sua approssimazione). La successione generata sarà

$$z_{k+1} = z_k - L(\mu, z_k)^{-1}\nabla F_\mu(z_k)$$

Se sostituiamo $z_k = x_k$, $\mu = \mu_k$, allora avremo

$$\nabla F_{\mu_k}(x_k) = \mu_k(x_k - p_{\mu_k}(x_k)) \sim \mu_k(x_k - y_{k+1}) = \widehat{s}_k$$

e pertanto confrontando la relazione sopra con la formula del Teorema 2

$$y_{k+1} = x_k - \frac{1}{\mu_k}\widetilde{s}_k$$

si ottiene $L(\mu_k, x_k) \sim \mu_k$. Nel caso di descend-step, $x_{k+1} = y_{k+1}$, dunque

$$y_{k+2} = y_{k+1} - \frac{1}{\mu_{k+1}}\widetilde{s}_{k+1}$$

e in analogia con prima, si avrà

$$\mu_{k+1} \sim L(\mu_k, y_{k+1}) \qquad L(\mu_k, y_{k+1})(y_{k+1} - y_k) \sim \nabla F_{\mu_k}(y_{k+1}) - \nabla F_{\mu_k}(y_k)$$

Per calcolare μ_{k+1}, dunque, ci servirebbe conoscere $\nabla F_{\mu_k}(y_{k+1})$, ma dato che questo non è sempre possibile, cerchiamo un'altra strada.

La *Reversal Form* prevede di trovare due punti z_k e z_{k+1} per cui $p_{\mu_k}(x_i) = z_i$. Se $s_i \in \partial f(x_i)$, allora avremo

$$x_i = z_i - \frac{1}{\mu_k}s_k \qquad \nabla F_{\mu_k}(z_i) = \mu_k(z_i - x_i) = s_i$$

$$\begin{aligned} v = {} & \nabla F_{\mu_k}(z_{k+1}) - \nabla F_{\mu_k}(z_k) && = s_{k+1} - s_k \\ u = {} & z_{k+1} - z_k && = x_{k+1} - x_k + \frac{1}{\mu_k}(s_{k+1} - s_k) \end{aligned}$$

Ricordando che $y_i = x_i$ in questo caso, e sostituendo sopra, si ottiene

$$\mu_{k+1} \sim \frac{\|v\|^2}{< v, u >} \implies \frac{1}{\mu_{k+1}} = \frac{1}{\mu_k} + \frac{< x_{k+1} - x_k, s_{k+1} - s_k >}{\|s_{k+1} - s_k\|^2}$$

Notiamo che μ_{k+1} non è sempre positivo, e per questo, cambieremo μ_k solo in caso di descend step, e solo quando $< u, v >$ risulti positivo. Tutto questo porta alla definizione della funzione *updpar*$(f, \mu_k, y_{k+1}, y_k, x_k)$

if $x_{k+1} = y_{k+1}$ **then**
Calcola $s_{k+1} \in \partial f(x_{k+1})$ e $s_k \in \partial f(x_k)$
if $< x_{k+1} - x_k, s_{k+1} - s_k > 0$ **then**
$1/\mu_{k+1} = 1/\mu_k + < x_{k+1} - x_k, s_{k+1} - s_k > / \|s_{k+1} - s_k\|^2$
end if
end if

Con questo, abbiamo finalmente descritto completamente l'algoritmo (GB). Vediamo ora alcuni risultati di convergenza.

2.4 Analisi di Convergenza

Notiamo che il fatto di aggiornare la funzione $\varphi_k(y)$ tramite la procedura *updmod*, fa uscire l'algoritmo (GB) da quelli descrivibili dall'algoritmo (CPG), poiché gli U^k vengono modificati ad ogni passo da funzioni diverse da H. Questo fa sì che il risultato di convergenza precedente non valga più, e pertanto ne dobbiamo trovare degli altri.

Se f non ha minimo, succede che $f(x_k)$ tendono a $-\infty$, dunque mettiamoci nel caso in cui il problema ha una soluzione ottimale.

In ogni caso, l'algoritmo (GB) genera una sequenza x_k discendente, dunque $f(x_k)$ converge a qualcosa. Notiamo che y_{k+1} potrebbe soddisfare $\delta_{k+1} \leq tol$, ma allo stesso tempo non rappresentare un descend-step, dunque non è detto che l'ultimo x_k soddisfi una relazione con tol.

Se $tol > 0$, allora l'algoritmo finisce in un numero finito di passi, pertanto fissiamo $tol = 0$, e dividiamo in due casi: se l'algoritmo genera infiniti descend-step o finiti. Nei teoremi che verranno, le costanti μ_k possono essere prese a piacimento, e non necessariamente aggiornati tramite *updpar*, dunque possiamo sceglierli a posteriori. Inoltre la costante m sarà presa nell'intervallo aperto $(0, 1)$.

Teorema 3. *Se (GB) genera infiniti descend-step (indicati con l'indice k per comodità), allora valgono*

1. $\{ \delta_k \} \to 0$, $\{ \varepsilon_k \} \to 0$
2. *Vale la freccia*

$$\sum_k \frac{1}{\mu_k} = +\infty \implies f(x_k) \to c = \min f(x)$$

3. *Se oltre a 2., esiste una costante d per cui $\mu_k \geq d > 0$ per ogni k, allora x_k convergeranno ad un minimo di $f(x)$.*

Proof. Innanzitutto, notiamo che gli x_k sono generati da descend-step, e se chiamiamo δ_k l'errore relativo a x_k, allora

$$f(x_k) - f(x_{k+1}) \geq m\delta_{k+1} \implies m \sum_{k=1}^{k_{max}} \delta_k \leq f(1) - f(x_{k_{max}})$$

ma dato che $f(x_k)$ converge ad un valore finito, allora anche la sommatoria degli errori convergerà. Con questo, avremo che

1. $\delta_k \to 0$, e di conseguenza $0 \leq \varepsilon_k \leq \delta_{k+1} \implies \varepsilon_k \to 0$

2. Dato un $x \in \mathbb{R}^n$, avremo

$$\begin{aligned}
\|x_{k+1} - x\|^2 &= \|x_k - x - \frac{1}{\mu_k}\widehat{s}_k\|^2 \\
&= \|x_k - x\|^2 + \frac{2}{\mu_k} < \widehat{s}_k, x - x_k > + \frac{1}{\mu_k^2}\|\widehat{s}_k\|^2 \\
&\leq \|x_k - x\|^2 + \frac{2}{\mu_k}(f(x) - f(x_k) + \varepsilon_k) + \frac{2}{\mu_k}\delta_{k+1} - \frac{2}{\mu_k}\varepsilon_k \\
&= \|x_k - x\|^2 + \frac{2}{\mu_k}(f(x) - f(x_k) + \delta_{k+1})
\end{aligned}$$

Se $\widetilde{x}$ è un minimo di f , e per assurdo esiste $\rho > 0$ per cui $f(x_k) \downarrow c \geq f(\widetilde{x}) + \rho$, allora esisterà $\delta_{k+1} < \rho/2 \;\; \forall\, k \geq k_\rho$, e sostituendo $x = \widetilde{x}$ sopra,

$$0 < \|x_{k+1} - \widetilde{x}\|^2 \leq \|x_k - \widetilde{x}\|^2 + \frac{2}{\mu_k}(f(\widetilde{x}) - f(x_k) + \delta_{k+1}) \leq \|x_k - \widetilde{x}\|^2 - \rho/\mu_k$$

$$0 < \|x_{k+1} - \widetilde{x}\|^2 \leq \|x_{k_\rho} - \widetilde{x}\|^2 - \rho \sum_{h=k_\rho}^{k} \mu_h \to -\infty$$

3. Riprendendo la formula sopra,

$$\begin{aligned}
0 < \|x_{k+1} - \widetilde{x}\|^2 &\leq \|x_1 - \widetilde{x}\|^2 + \sum_{h=1}^{k} \frac{2}{\mu_h}(f(\widetilde{x}) - f(x_h) + \delta_{h+1}) \\
&\leq \|x_1 - \widetilde{x}\|^2 + \frac{2}{d}\sum_{h=1}^{k}(f(\widetilde{x}) - f(x_h)) + \frac{2}{d}\sum_{h=1}^{k}\delta_{h+1}
\end{aligned}$$

ma $f(\widehat{x}) - f(x_h) \leq 0$, dunque la sommatoria converge, e la successione $\|x_{k+1} - \widetilde{x}\|^2$ sarà limitata. Questo ci dice che i x_k sono in un compatto, e pertanto esisterà una sottosuccessione convergente, e dato che siamo nelle ipotesi di 2., dovrà convergere ad un minimo, che supponiamo essere $\widetilde{x}$. Per dimostrare che tutta la serie converge, notiamo che per ogni q, p vale

$$0 < \|x_{q+1} - \widetilde{x}\|^2 \leq \|x_p - \widetilde{x}\|^2 + \frac{2}{d}\sum_{h=p}^{q}\delta_{h+1}$$

e se scegliamo p grande e appartenente alla sottosuccessione, riusciamo a limitare questa quantità con un ε piccolo a piacere.

□

Questo teorema, unito ad un'opportuna scelta degli μ_k (per esempio costanti, o $\mu_k = k$), ci dice che l'algoritmo converge, e che la sottosuccessione degli x_k è una successione discendente che tende ad un punto minimo $\bar{x}$.

Prendere però μ_k non limitati non è ottimale, in quanto

Teorema 4. *Se (GB) genera finiti descend-step, di cui l'ultimo k_{last} ha generato x_{last}, e poi infiniti null-step, e se*

1. $\mu_{k+1} \geq \mu_k \ \ \forall k \geq k_{last}$
2. $\exists\, C > 0 \mid \mu_k \leq C \ \ \forall k$

allora y_k convergono a x_{last}, che è un minimo di $f(x)$

Proof. D'ora in poi indicheremo con x il punto x_{last}, e tutti gli indici k che useremo saranno $k \geq k_{last}$. Ricordiamo le relazioni

$$\widehat{s}_k = \mu_k(x - y_{k+1}) \qquad y_{k+1} = \min\left(\varphi_k(y) + \frac{1}{2}\mu_k\|y - x\|^2\right)$$

$$\delta_{k+1} = \varepsilon_k + \frac{\|\widehat{s}_k\|^2}{2\mu_k} = f(x) - \varphi_k(y_{k+1}) - \frac{1}{2}\mu_k\|y_{k+1} - x\|^2$$

$$\widetilde{f}_k(y) = \varphi_k(y_{k+1}) + < \widehat{s}_k, y - y_{k+1} > = f(x) - \varepsilon_k + < \widehat{s}_k, y - x >$$

e notiamo che in ogni caso, vale la relazione

$$\varphi_{k+1}(y) \geq \widetilde{f}_k(y) \ \ \forall\, y$$

Introduciamo dunque la funzione

$$L_k(y) = \varphi_k(y_{k+1}) + \frac{1}{2}\mu_k\|y_{k+1} - x\|^2 + \frac{1}{2}\mu_k\|y_{k+1} - y\|^2$$

Notiamo che

$$\begin{aligned} L_k(y_{k+1}) + \frac{1}{2}\mu_k\|y_{k+1} - x\|^2 &= \varphi_k(y_{k+1}) + \mu_k\|y_{k+1} - x\|^2 \\ &= \varphi_k(y_{k+1}) + < \widehat{s}_k, x - y_{k+1} > = \widetilde{f}_k(x) \leq f(x) \end{aligned}$$

Calcoliamo ora

$$\begin{aligned} L_{k+1}(y_{k+2}) &= \varphi_{k+1}(y_{k+2}) + \frac{1}{2}\mu_{k+1}\|y_{k+2} - x\|^2 \\ &\geq \widetilde{f}_k(y_{k+2}) + \frac{1}{2}\mu_k\|y_{k+2} - x\|^2 \\ &= \varphi_k(y_{k+1}) + < \widehat{s}_k, y_{k+2} - y_{k+1} > + \frac{1}{2}\mu_k\|y_{k+2} - x\|^2 \\ &= L_k(y_{k+1}) + < \widehat{s}_k, y_{k+2} - y_{k+1} > + \frac{1}{2}\mu_k\left(\|y_{k+2} - x\|^2 - \|y_{k+1} - x\|^2\right) \end{aligned}$$

in particolare

$$\begin{aligned}\|y_{k+2}-x\|^2-\|y_{k+1}-x\|^2 &= \ < y_{k+2}+y_{k+1}-2x, y_{k+2}-y_{k+1} > \\ &= \|y_{k+2}-y_{k+1}\|^2+2 < y_{k+1}-x, y_{k+2}-y_{k+1} > \\ &= \|y_{k+2}-y_{k+1}\|^2-\frac{2}{\mu_k} < \widehat{s}_k, y_{k+2}-y_{k+1} >\end{aligned}$$

sostituendo dentro si ottiene

$$L_{k+1}(y_{k+2}) \geq L_k(y_{k+1})+\frac{1}{2}\mu_k\|y_{k+2}-y_{k+1}\|^2 \geq L_k(y_{k+1})$$

dunque $\{\, L_k(y_{k+1})\, \}$ è una successione non decrescente in k, e dato che

$$L_k(y_{k+1}) \leq L_k(y_{k+1})+\frac{1}{2}\mu_k\|y_{k+1}-x\|^2 \leq f(x)$$

è anche limitata, e pertanto converge. Dato che μ_k è limitata dal basso, allora avremo anche che $\|y_{k+2}-y_{k+1}\|$ va a zero, e che $\|y_{k+1}-x\|$ è limitata, dunque gli y_k sono contenuti in un compatto.

Avendo preso $f(x)$ convessa, sappiamo che è Lipschitziana, con sottodifferenziale limitato sui compatti, e se ne prendiamo uno che contiene definitivamente tutti gli y_k, allora esistono L, M per cui

$$-M\|y_{k+1}-y_k\| \ \leq \ < s_{k+1}, y_{k+1}-y_k > \ \leq \ \varphi_k(y_{k+1})-f(y_k)$$

$$\varphi_k(y_{k+1})-f(y_k) \leq f(y_{k+1})-f(y_k) \leq L\|y_{k+1}-y_k\|$$

ma sappiamo già che $\|y_{k+1}-y_k\|$ va a zero, dunque anche $\varphi_k(y_{k+1})-f(y_k)$ e $f(y_{k+1})-f(y_k)$ vanno a zero. In particolare andrà a zero anche

$$f(y_{k+1})-\varphi_k(y_{k+1}) = \big[f(y_{k+1})-f(y_k)\big]-\big[f(y_{k+1})-f(y_k)\big]$$

Avere un numero infinito di null-step implica che per ogni k si ha

$$f(x)-f(y_{k+1}) < m\delta_{k+1} \qquad 0 < \delta_{k+1}$$

$$\begin{aligned}0 \leq (1-m)\delta_{k+1} &< \delta_{k+1}+f(y_{k+1})-f(x) \\ &= f(y_{k+1})-\varphi_k(y_{k+1})-\frac{1}{2}\mu_k\|y_{k+1}-x\|^2 \\ &\leq f(y_{k+1})-\varphi_k(y_{k+1}) \to 0\end{aligned}$$

Dunque anche l'errore δ_k tende a zero, e ciò implica che anche ε_k e $\widehat{s}_k$ tendono a zero, perché i μ_k sono limitati dall'alto. Ma andando al limite otteniamo

$$\widehat{s}_k \in \partial_{\varepsilon_k} f(x) \implies 0 \in \partial f(x) \qquad \|\widehat{s}_k\| = \mu_k\|y_{k+1}-x\| \implies y_k \to x$$

Dunque x è minimo di $f(y)$, e gli y_k tendono a x.

$\square$

Anche qua, la scelta di μ_k costanti porta alla convergenza ad un minimo in finiti passaggi.

3 Conclusioni

L'algoritmo Bundle porta vantaggi sia per la stabilità del metodo, che per la complessità, poiché ad ogni passo riusciamo a risolvere dei problemi con al massimo n_{max} vincoli.

Inoltre, questo metodo ha un po' di gradi di libertà, che è possibile gestire in modo da migliorare la velocità e l'approssimazione. Per esempio

- la scelta dei μ_k
- la scelta di tol, tol_ε e $tol_{\widehat{s}}$
- la scelta di m
- la scelta di una metrica diversa da quella euclidea.

L'ultima opzione, in particolare, ci permette di ridefinire i prodotti scalari e le norme con opportune matrici simmetriche, che è possibile aggiornare passaggio per passaggio, e portano a diversi teoremi di convergenza.