Programmazione non-lineare

Giovanni Righini

Ricerca Operativa

Università degli Studi
di Milano

Programmazione non-lineare (PNL)

La programmazione non-lineare, o PNL (Non-linear Programming, NLP) studia problemi di ottimizzazione in cui la funzione obiettivo o alcuni vincoli sono non-lineari.

Applicazioni:

- economie di scala,
- minimizzazione dell'errore quadratico medio in problemi di
  - controllo ottimo,
  - classificazione automatica,
  - machine learning,
  - fitting di dati sperimentali,
- riformulazioni quadratiche,
- modelli di sistemi fisici non lineari,
- modelli che implicano l'uso di distanza Euclidee,
- eccetera...

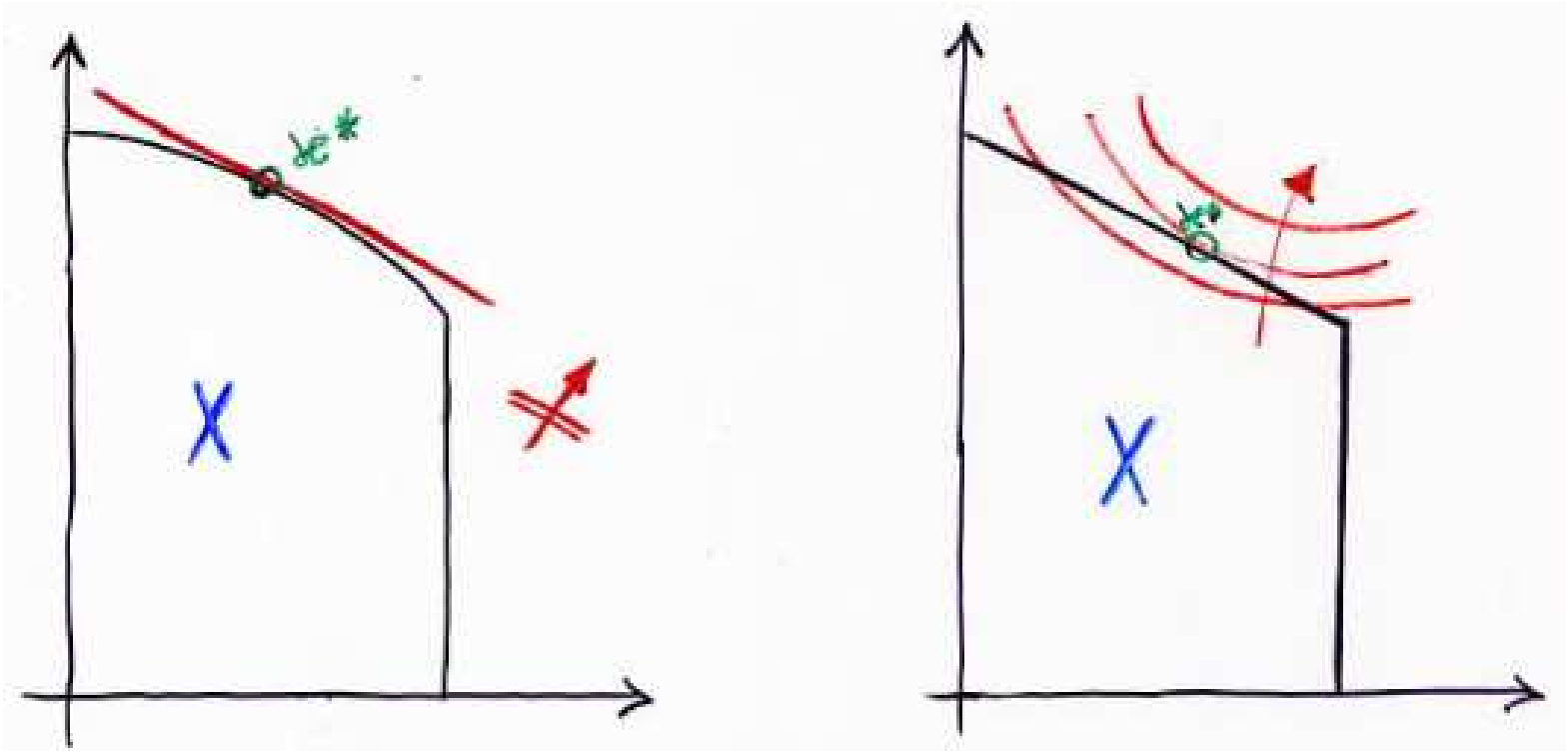
Programmazione non-lineare (PNL)

Forma generale:

$$\begin{aligned} \text{minimize } z = & f(x) \\ \text{s.t. } & h_i(x) = 0 \qquad \forall i \\ & g_j(x) \leq 0 \qquad \forall j \\ & x \in \Re^n \end{aligned} \tag{1}$$

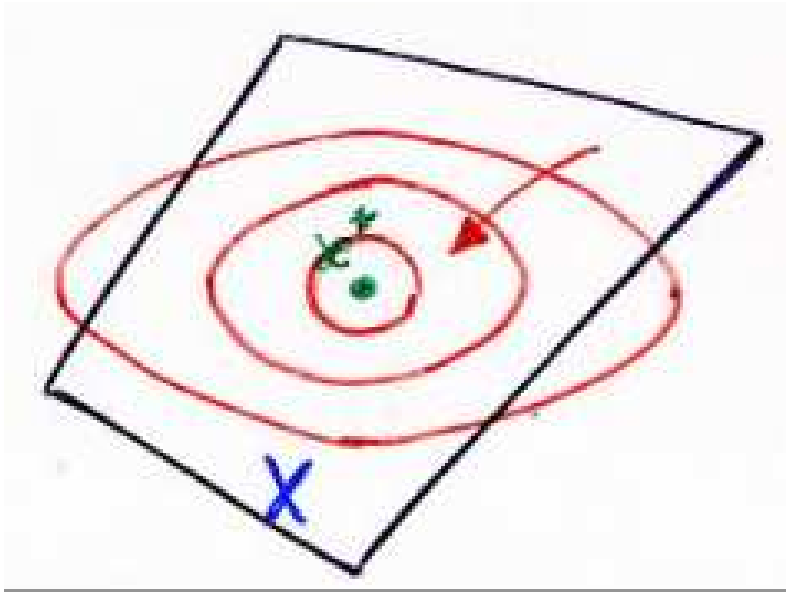
dove $f(x)$, $g(x)$ e $h(x)$ possono essere funzioni non-lineari.

Programmazione non-lineare (PNL)



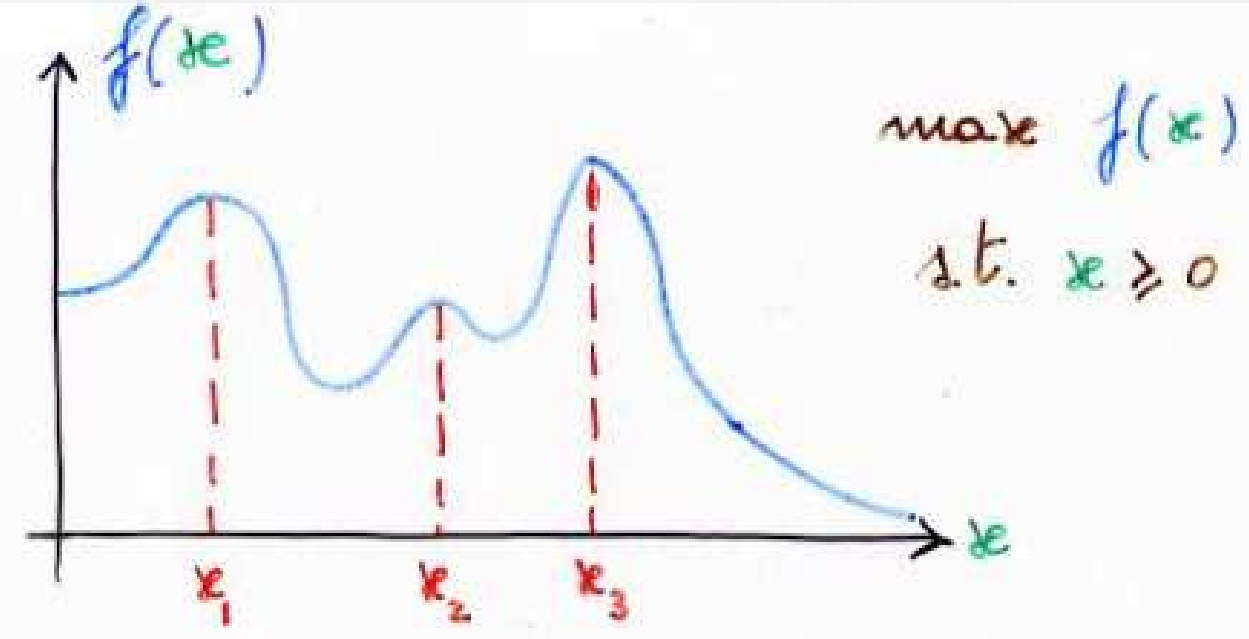
In generale, la soluzione ottima può non essere all'intersezione dei vincoli.

Programmazione non-lineare (PNL)



Non è neppure detto che sia necessariamente sulla frontiera della regione ammissibile.

Ottimalità locale e globale



Le soluzioni x_1 e x_2 sono ottimi locali.
La soluzione x_3 è un ottimo globale.

Ottimalità locale e globale

Ottimalità globale. Una soluzione $x^* \in X$ è un minimo globale se e solo se

$$f(x^*) \leq f(x) \quad \forall x \in X.$$

Ottimalità locale. Una soluzione $\overline{x} \in X$ è un minimo locale se e solo se

$$\exists \epsilon > 0 : f(\overline{x}) \leq f(x) \quad \forall x \in X : ||\overline{x} - x|| \leq \epsilon.$$

L'insieme delle soluzioni $x \in X : ||\overline{x} - x|| \leq \epsilon$ è un intorno di $\overline{x}$.

Ottimalità locale e globale

Per trovare un ottimo globale si dovrebbero enumerare tutti gli ottimi locali e scegliere il migliore.

Tuttavia, l'enumerazione completa degli ottimi locali in generale non è fattibile in pratica

- per il loro grande numero;
- perché non è noto un metodo algoritmico per eseguirla in modo efficiente.

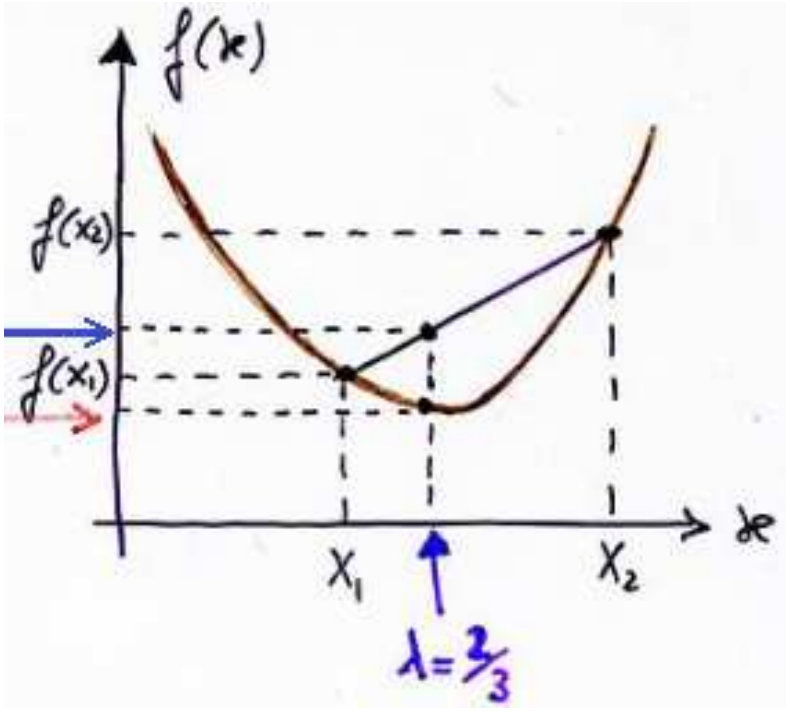
Un'importante eccezione positiva è la programmazione convessa. Un problema di minimizzazione non-lineare è convesso quando

- la funzione-obiettivo è una funzione convessa;
- la regione ammissibile è un insieme convesso.

Funzioni convesse

Una funzione $f(x)$ è convessa se e solo se per ogni coppia di punti x_1 e x_2 nel suo dominio e per $0 \leq \lambda \leq 1$

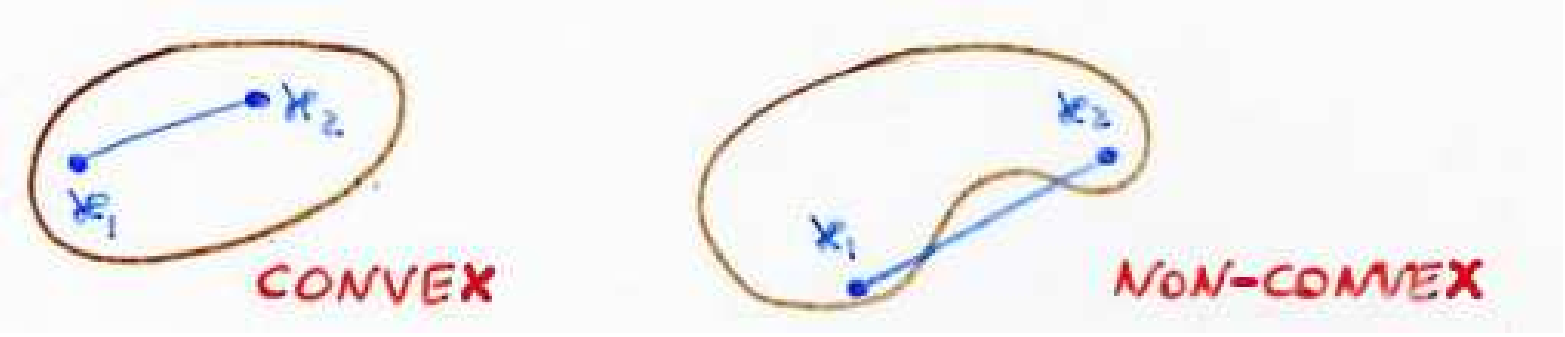
$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$



Insiemi convessi

Un insieme X è convesso se e solo se per ogni coppia di punti x_1 e x_2 in esso, tutte le loro combinazioni convesse appartengono all'insieme:

$$\forall x_1, x_2 \in X \quad \forall 0 \leq \lambda \leq 1 \quad \lambda x_1 + (1 - \lambda) x_2 \in X .$$



Programmazione convessa

La regione ammissibile è convessa quando

- tutti i vincoli di uguaglianza $h(x) = 0$ sono lineari;
- tutti i vincoli di disuguaglianza, riscritti in forma $g(x) \leq 0$ sono convessi.

La funzione-obiettivo da minimizzare deve essere convessa (deve essere concava in caso di massimizzazione).

Se entrambe queste condizioni sono soddisfatte, il problema è di programmazione convessa e quindi:

- l'ottimalità locale implica quella globale;
- se esistono più ottimi, essi formano un insieme convesso.

Ottimizzazione vincolata e non vincolata

Distinguiamo tra

- Unconstrained NLP: minimizzare una funzione non lineare senza ulteriori vincoli.
- Constrained NLP: minimizzare $f(x)$, con $x \in X$: le non-linearità possono essere tanto nell'obiettivo quanto nei vincoli.

Ottimizzazione non vincolata

Assumiamo che la funzione-obiettivo $f(x)$ da minimizzare sia continua e differenziabile.

Il gradiente di una funzione $f(x_1, x_2, \ldots, x_n)$ è il vettore delle sue derivate parziali di primo ordine

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1} \; \frac{\partial f}{\partial x_2} \; \cdots \; \frac{\partial f}{\partial x_n} \right]^T .$$

L'Hessiano di una funzione $f(x_1, x_2, \ldots, x_n)$ è la matrice delle sue derivate parziali di secondo ordine

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} .$$

Caratterizzazione dei minimi locali

Condizioni necessarie del primo ordine.

$$\nabla f(\overline{x}) = 0$$

Condizioni necessarie del secondo ordine.

$$\nabla^2 f(\overline{x}) \geq 0$$

Condizioni sufficienti del secondo ordine.

$$\nabla^2 f(\overline{x}) > 0$$

Algoritmi

Se le derivate prime e seconde sono note (il che non è garantito, in generale), si possono enumerare i punti nei quali sono soddisfatte le condizioni analitiche.

Gli algoritmi per l'ottimizzazione non-lineare sono algoritmi iterativi, che convergono verso un minimo locale.
Partono da una soluzione data $x^{(0)}$ e calcolano una sequenza di soluzioni tali che il valore di $f(x)$ diminuisce monotonicamente.
Si fermano quando il miglioramento ottenuto o il passo compiuto sono più piccoli di una data soglia.

Ad ogni iterazione k, l'algoritmo calcola una direzione $d^{(k)}$ (vettore) e un passo s_k (scalare) tali che:

$$x^{(k+1)} = x^{(k)} + s_k d^{(k)}.$$

Le due principali strategie sono:

- line search;
- trust regions.

Algoritmi *line search*

Negli algoritmi *line search*, le scelte più comuni per definire la direzione $d^{(k)}$ sono:

- (metodo del gradiente): la direzione opposta a quella del gradiente, $-\nabla f(x^{(k)})$;
- (metodo di Newton): una direzione $-B^{-1}\nabla f(x^{(k)})$, dove B è una matrice semi-definita positiva;
- (metodo del gradiente coniugato): una direzione $-\nabla f(x^{(k)}) + \beta_k d^{(k-1)}$.

Metodo del gradiente

Per il teorema di Taylor

$$f(x^{(k)}+s_k d^{(k)}) = f(x^{(k)})+s_k d^{(k)T}\nabla f(x^{(k)})+\frac{1}{2}s_k^2 d^{(k)^T}\nabla^2 f(x^{(k)})d^{(k)}+\dots$$

Trascurando i termini dal secondo ordine in poi, si ha l'approssimazione

$$f(x^{(k)} + s_k d^{(k)}) = f(x^{(k)}) + s_k d^{(k)T}\nabla f(x^{(k)})$$

che decresce più rapidamente nella direzione opposta a quella del *gradiente*.

$$d^{(k)} = -\frac{\nabla f(x^{(k)})}{||\nabla f(x^{(k)})||}.$$

Un vantaggio di questo metodo, detto *steepest descent method* (o *gradient method*) è che richiede solo il calcolo del gradiente, non delle derivate seconde.

Metodo di Newton

Assumendo $s_k = 1$ e trascurando i termini dal terzo ordine in poi, si ha l'approssimazione

$$f(x^{(k)} + d^{(k)}) = f(x^{(k)}) + d^{(k)T} \nabla f(x^{(k)}) + \frac{1}{2} d^{(k)T} \nabla^2 f(x^{(k)}) d^{(k)}.$$

La direzione che minimizza questa quantità è la *direzione di Newton*:

$$d^{(k)} = -\nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}).$$

Il *metodo di Newton* è veloce e accurato, ma richiede il calcolo dell'Hessiano $\nabla^2 f(x^{(k)})$ e può essere usato solo quando $\nabla^2 f(x^{(k)})$ è definito positivo.

Metodi *quasi-Newton* sono stati ideati per ovviare a questo limite.

Scelta del passo

Una volta scelta la direzione $d^{(k)}$, rimane un problema di minimizzazione ad una sola variabile

$$\text{minimize } f(x^{(k+1)}) = f(x^{(k)} + s_k d^{(k)})$$

dove la variabile è lo scalare $s_k \geq 0$.

L'ottimizzazione esatta di s_k non è indispensabile; una buona approssimazione è sufficiente per avviare l'iterazione successiva dopo aver migliorato $f(x)$.

Gli algoritmi per determinare il passo possono essere classificati in

- algoritmi che richiedono il calcolo della derivata,
- algoritmi *derivative-free*.

Metodo di bisezione

Richiede il calcolo della derivata.

Dato un intervallo iniziale $r = [a, b]$ per s_k:

1. calcolare $\nabla f(x^{(k)} + \frac{a+b}{2} d^{(k)})$;
2. se è positiva, porre $r := [a, \frac{a+b}{2}]$;
3. se è negativa, porre $r := [\frac{a+b}{2}, b]$;
4. ripetere finché r è abbastanza piccolo.

I numeri di Fibonacci

La sequenza dei numeri di Fibonacci inizia con $F_0 = 0$ e $F_1 = 1$ e si ricava applicando la ricorsione

$$F_k = F_{k-1} + F_{k-2}.$$

Si ottiene così:

k	0	1	2	3	4	5	6	7	8	9	10	11	...
F_k	0	1	1	2	3	5	8	13	21	34	55	89	...

Tabella: I primi numeri di Fibonacci.

Una proprietà

Proprietà. Dati 4 numeri di Fibonacci consecutivi a partire dal k-esimo, si ha

$$F_{k+1}F_{k+2} - F_kF_{k+3} = (-1)^k \quad \forall k \geq 0.$$

Esempio.

$$F_6F_7 - F_5F_8 = 8 \times 13 - 5 \times 21 = 104 - 105 = -1 \quad (k = 5, \text{dispari})$$

$$F_7F_8 - F_6F_9 = 13 \times 21 - 8 \times 34 = 273 - 272 = 1 \quad (k = 6, \text{pari}).$$

Dimostrazione (per induzione).

$$F_{k+1}F_{k+2} - F_kF_{k+3} = (-1)^k \quad \forall k \geq 0.$$

- Base dell'induzione (per $k = 0$):

$$F_1F_2 - F_0F_3 = 1 \times 1 - 0 \times 2 = 1^0$$

- Passo induttivo (da $k-1$ a k per ogni $k \geq 1$:

$$F_kF_{k+1} - F_{k-1}F_{k+2} = (-1)^{k-1} \Rightarrow F_{k+1}F_{k+2} - F_kF_{k+3} = (-1)^k.$$

Infatti:

$$\begin{aligned} F_{k+1}F_{k+2} - F_kF_{k+3} &= [F_{k+1}(F_k + F_{k+1})] - [F_k(2F_{k+1} + F_k)] = \\ &= F_kF_{k+1} + F_{k+1}^2 - 2F_kF_{k+1} - F_k^2 = (F_{k+1}^2 - F_k^2) - F_kF_{k+1} = \\ &= (F_{k+1} + F_k)(F_{k+1} - F_k) - F_kF_{k+1} = F_{k+2}F_{k-1} - F_kF_{k+1} = \text{ (per ipotesi)} \\ &= -(-1)^{k-1} = (-1)^k. \quad \text{[c.v.d.]} \end{aligned}$$

Il problema

- È data una funzione continua $f(x)$ di una sola variabile x.
- Si vuole cercare un punto di minimo di $f(x)$.
- È dato un intervallo di incertezza iniziale I^0.
- È richiesto un massimo intervallo di incertezza finale Δ.
- Si assume che la funzione sia *unimodale* nell'intervallo I^0, cioè abbia un solo punto di minimo nell'intervallo.
- Si suppone di non poter/voler calcolare la derivata prima di $f(x)$, come invece è richiesto dal metodo di bisezione.
- Si suppone che sia possibile valutare $f(x)$ in punti diversi, purché distanti tra loro almeno ϵ (risoluzione).

Osservazione. Sarebbe ancora possibile usare il metodo di bisezione, valutando in ogni intervallo due punti distanti ϵ tra loro, posti al centro dell'intervallo stesso. L'informazione che se ne ricaverebbe sarebbe equivalente a quella fornita dal calcolo della derivata prima al centro dell'intervallo. In tal modo sarebbero necessarie *due* valutazioni della funzione ad ogni iterazione. Con il metodo dei numeri di Fibonacci, invece, basta *una* valutazione della funzione ad ogni iterazione.

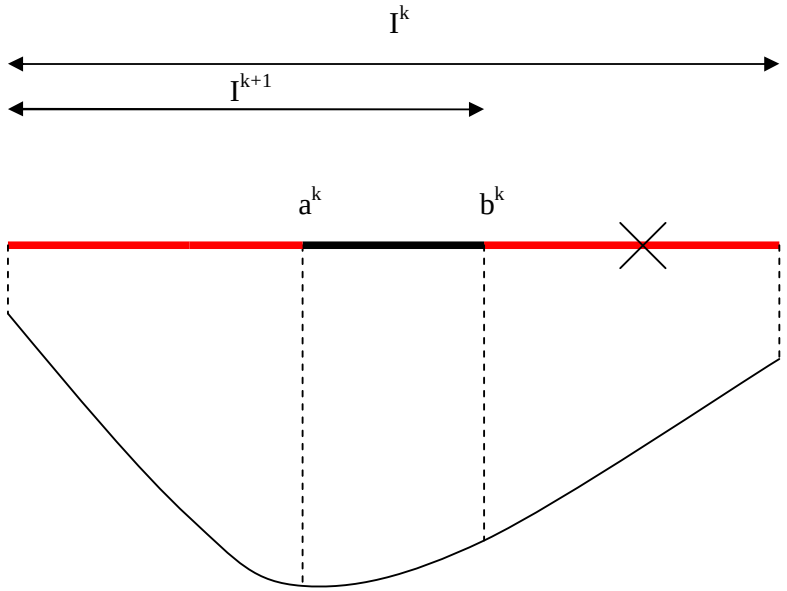
Iterazione generica

Alla generica iterazione k si ha un intervallo di incertezza I^k.

Si considerino due punti a^k e b^k interni all'intervallo, che lo dividono in tre parti.
Si conosca il valore della funzione $f(x)$ nei due punti interni.

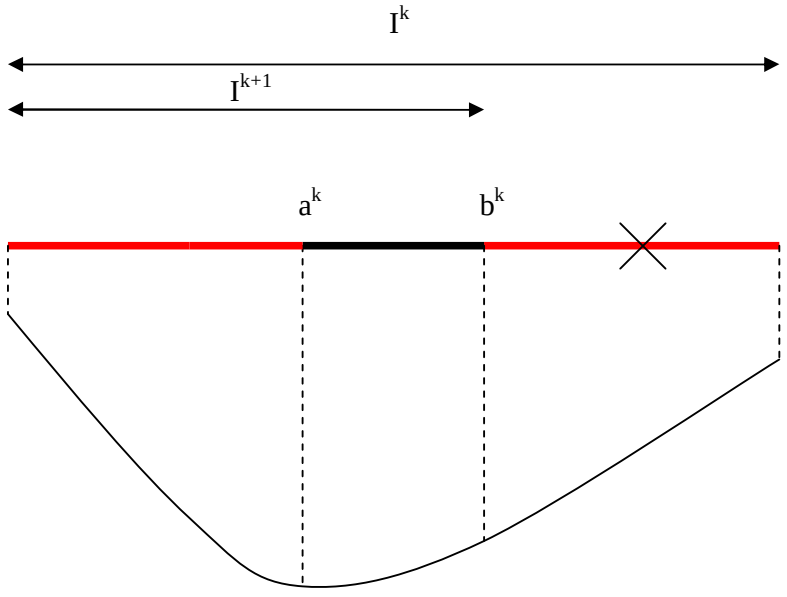
Se $f(a^k) > f(b^k)$, allora il minimo di $f(x)$ non cade nella prima parte.

Se $f(a^k) < f(b^k)$, allora il minimo di $f(x)$ non cade nella terza parte.



Punti di valutazione

Alla successiva iterazione l'intervallo di incertezza risulta composto da due delle tre parti dell'intervallo di incertezza precedente.

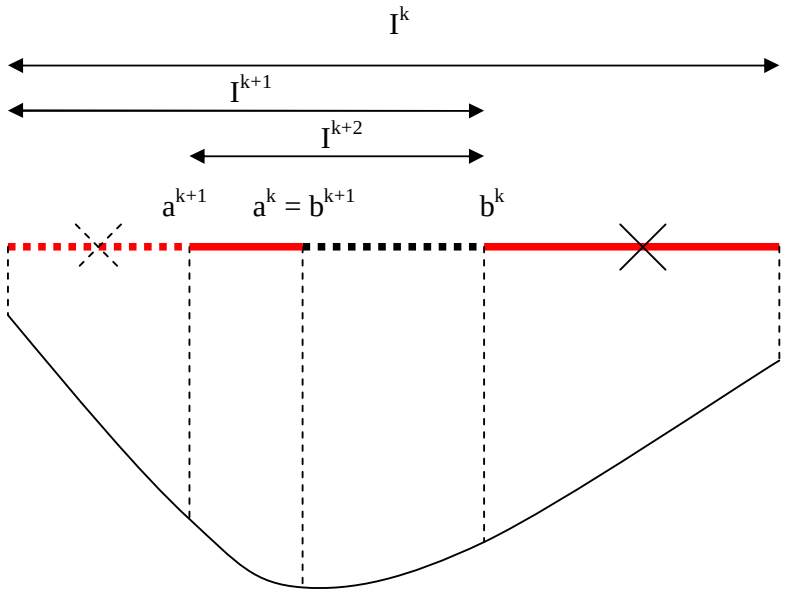


Uno dei due punti interni precedenti diventa un estremo dell'intervallo di incertezza.

Punti di valutazione

Per simmetria, ad ogni iterazione i punti in cui valutare $f(x)$ sono scelti in modo simmetrico nell'intervallo di incertezza corrente.
Si ha quindi:

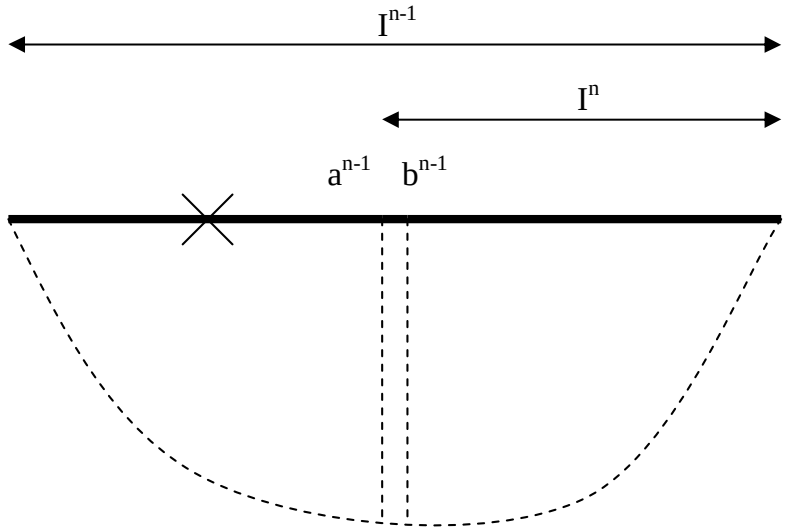
$$I^k = I^{k+1} + I^{k+2} \quad \forall k \geq 0.$$



In uno dei due punti interni la funzione è già stata valutata in precedenza.

Intervallo finale

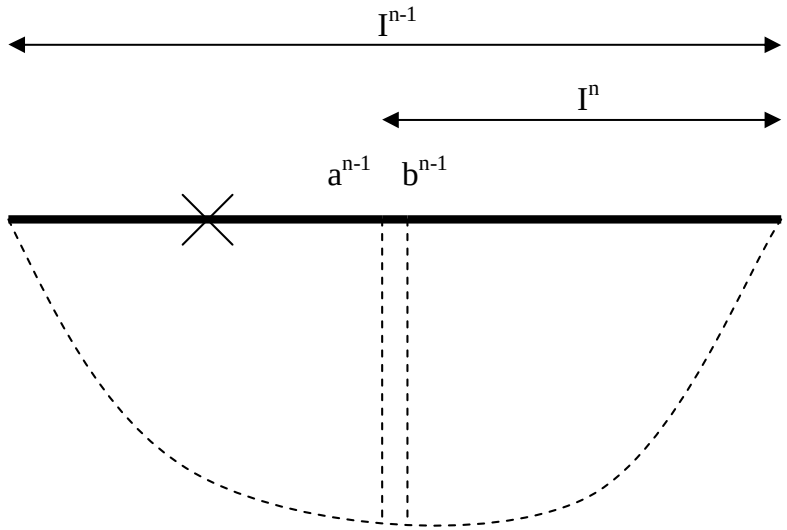
Osservazione. Più è grande l'intervallo "scartato" all'iterazione k e più risultano piccoli quelli "scartabili" all'iterazione $k + 1$.
L'iterazione di massima efficacia è quella che consente di scartare metà dell'intervallo di incertezza, valutando due punti interni distinti vicinissimi tra loro (a distanza ϵ).



Tuttavia all'iterazione successiva il primo e il terzo intervallo sarebbero larghi solo ϵ e quindi si avrebbe un'iterazione di minima efficacia.

Intervallo finale

Si vuole quindi arrivare a compiere un'iterazione di massima efficacia *per ultima*.



Si ha perciò:

$$l^{n-1} = 2l^{n} - \epsilon.$$

Progressione delle iterazioni

Dalle due relazioni

$$I^k = I^{k+1} + I^{k+2} \quad \forall k \geq 0$$

$$I^{n-1} = 2I^n - \epsilon$$

si ricava:

$$I^{n-2} = I^{n-1} + I^n = 3I^n - \epsilon$$

$$I^{n-3} = I^{n-2} + I^{n-1} = 5I^n - 2\epsilon$$

$$I^{n-4} = I^{n-3} + I^{n-2} = 8I^n - 3\epsilon$$

$$\dots$$

$$I^{n-k} = I^{n-k+1} + I^{n-k+2} = F_{k+2}I^n - F_k\epsilon \quad \forall k \geq 1$$

$$\dots$$

$$I^2 = I^3 + I^4 = F_nI^n - F_{n-2}\epsilon$$

$$I^1 = I^2 + I^3 = F_{n+1}I^n - F_{n-1}\epsilon$$

$$I^0 = I^1 + I^2 = F_{n+2}I^n - F_n\epsilon.$$

Perciò:

$$I^0 = F_{n+2}I^n - F_n\epsilon \text{ ovvero } I^n = \frac{I^0}{F_{n+2}} + \frac{F_n}{F_{n+2}}\epsilon.$$

Numero di iterazioni

Dalla relazione

$$I^n = \frac{I^0}{F_{n+2}} + \frac{F_n}{F_{n+2}}\epsilon$$

e dal requisito sull'incertezza finale

$$I^n \leq \Delta$$

si ricava il numero di iterazioni necessarie:

$$\overline{n} = \min\{n | \frac{I^0}{F_{n+2}} + \frac{F_n}{F_{n+2}}\epsilon \leq \Delta\}.$$

Scelta del primo punto interno

Dalle relazioni

$$I^{\overline{n}} = \frac{I^0}{F_{\overline{n}+2}} + \frac{F_{\overline{n}}}{F_{\overline{n}+2}}\epsilon$$

e

$$I^1 = F_{\overline{n}+1} I^{\overline{n}} - F_{\overline{n}-1}\epsilon$$

si ricava I^1 in funzione di I^0:

$$\begin{aligned} I^1 &= F_{\overline{n}+1}\left(\frac{I^0}{F_{\overline{n}+2}} + \frac{F_{\overline{n}}}{F_{\overline{n}+2}}\epsilon\right) - F_{\overline{n}-1}\epsilon = \\ &= \frac{F_{\overline{n}+1}}{F_{\overline{n}+2}} I^0 + \frac{\epsilon}{F_{\overline{n}+2}}\left(F_{\overline{n}+1}F_{\overline{n}} - F_{\overline{n}+2}F_{\overline{n}-1}\right) = \\ &= \frac{F_{\overline{n}+1}}{F_{\overline{n}+2}} I^0 + \frac{\epsilon}{F_{\overline{n}+2}}(-1)^{\overline{n}-1}. \end{aligned}$$

Esempio

- È dato un intervallo di incertezza iniziale $I^0 = [0, 100]$.
- È richiesto un massimo intervallo di incertezza finale $\Delta = 2$.
- È data una risoluzione $\epsilon = 1$.

Esempio

Scelta del numero di iterazioni

$$\overline{n} = \min\{n | \frac{100}{F_{n+2}} + \frac{F_n}{F_{n+2}} \leq 2\} = 9.$$

Infatti si ha:
per $n = 8$: $\frac{100}{F_{10}} + \frac{F_8}{F_{10}} = 100/55 + 21/55 = 121/55 > 2$,
per $n = 9$: $\frac{100}{F_{11}} + \frac{F_9}{F_{11}} = 100/89 + 34/89 = 134/89 < 2$.

k	0	1	2	3	4	5	6	7	8	9	10	11	...
F_k	0	1	1	2	3	5	8	13	21	34	55	89	...

Scelta del primo punto interno

$$I^1 = \frac{F_{\overline{n}+1}}{F_{\overline{n}+2}} I^0 + \frac{\epsilon}{F_{\overline{n}+2}} (-1)^{\overline{n}-1} =$$
$$= \frac{55}{89} 100 + \frac{1}{89} (-1)^8 = 5500/89 + 1/89 = 5501/89 \approx 61,81.$$

Esempio

Iterazioni. Gli intervalli di incertezza successivi risultano avere le seguenti ampiezze:

$$I^2 = I^0 - I^1 = \frac{8900 - 5501}{89} = \frac{3399}{89}$$

$$I^3 = I^1 - I^2 = \frac{5501 - 3399}{89} = \frac{2102}{89}$$

$$I^4 = I^2 - I^3 = \frac{3399 - 2102}{89} = \frac{1297}{89}$$

$$I^5 = I^3 - I^4 = \frac{2102 - 1297}{89} = \frac{805}{89}$$

$$I^6 = I^4 - I^5 = \frac{1297 - 805}{89} = \frac{492}{89}$$

$$I^7 = I^5 - I^6 = \frac{805 - 492}{89} = \frac{313}{89}$$

$$I^8 = I^6 - I^7 = \frac{492 - 313}{89} = \frac{179}{89}$$

$$I^9 = I^7 - I^8 = \frac{313 - 179}{89} = \frac{134}{89}.$$

Conclusioni

- Il metodo dei numeri di Fibonacci consente di approssimare il minimo di una funzione di una sola variabile continua.
- Deve essere noto un intervallo di incertezza iniziale e la funzione deve essere unimodale in esso.
- Il metodo non richiede il calcolo della derivata prima della funzione.
- Il metodo richiede di valutare la funzione in un numero di punti dello stesso ordine di grandezza del numero di iterazioni.

Ottimizzazione vincolata

Nell'ottimizzazione non-lineare vincolata consideriamo anche l'effetto di

- vincoli di uguaglianza $h_i(x) = 0 \; \forall i \in \mathcal{E}$
- vincoli di disuguaglianza $g_i(x) \leq 0 \; \forall i \in \mathcal{I}$.

Un vincolo di disuguaglianza $j \in \mathcal{I}$ è attivo in una soluzione $\overline{x}$ se e solo se $g_j(\overline{x}) = 0$.

Vincoli di uguaglianza

Consideriamo un vincolo di uguaglianza $h(x) = 0$ ed un punto $\overline{x}$ su di esso.
Indichiamo con $\nabla h(\overline{x})$ la direzione della normale al vincolo in $\overline{x}$.
Consideriamo un passo infinitesimo da $\overline{x}$ lungo una direzione d.
Per mantenere l'ammissibilità rispetto al vincolo, d deve essere tale che:

$$\nabla h(\overline{x})^T d = 0.$$

Il passo produce un miglioramento nel valore di $f(x)$ se e solo se

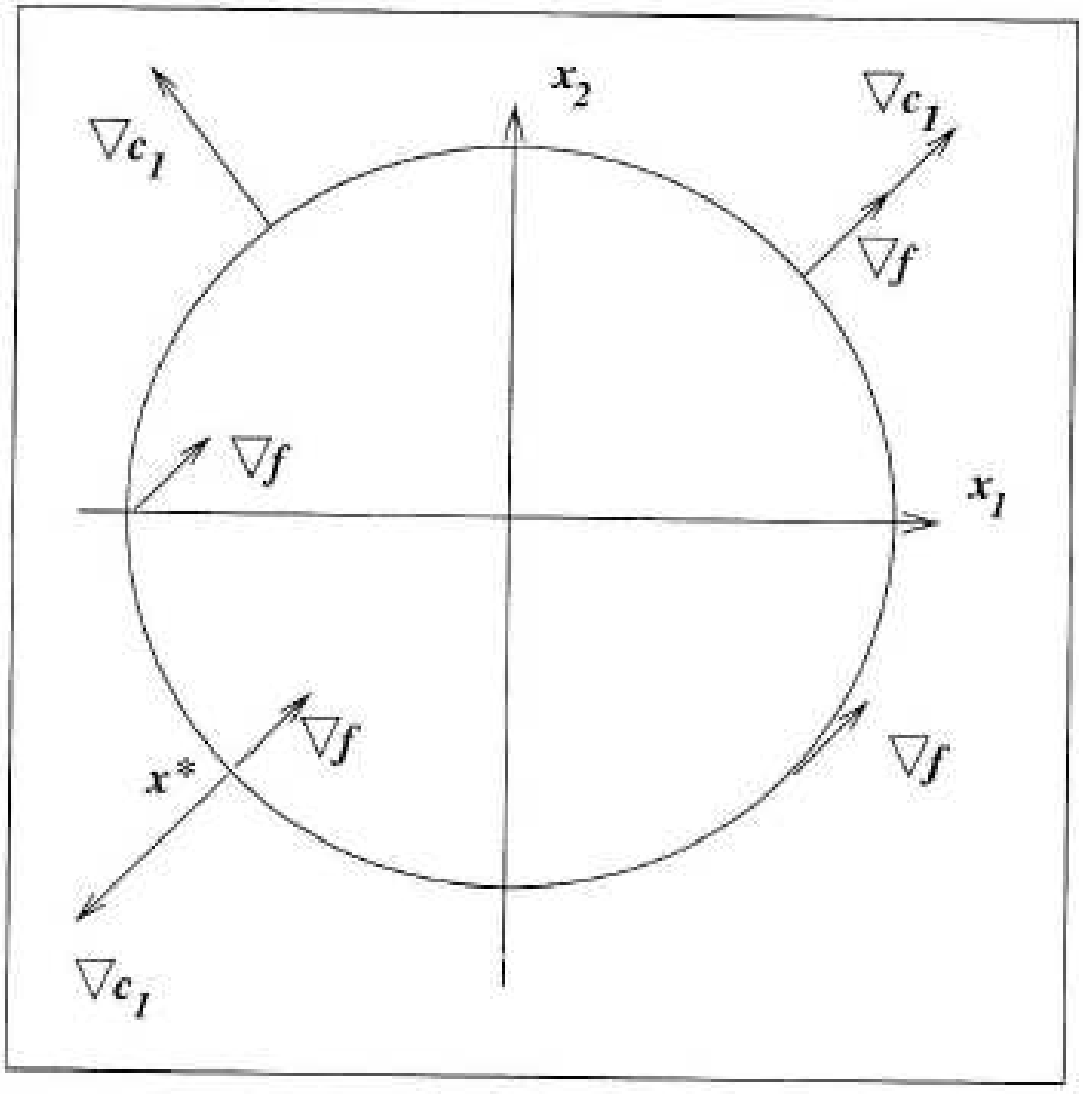
$$\nabla f(\overline{x})^T d < 0.$$

Qundi un passo migliorante *non* è possibile se

$$\nabla h(\overline{x}) = \lambda \nabla f(\overline{x})$$

per qualche $\lambda \neq 0$.

Vincoli di uguaglianza



Vincoli di disuguaglianza

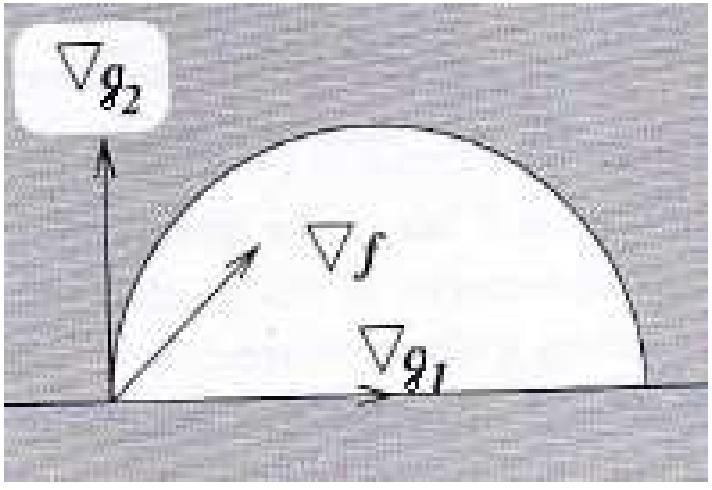
Consideriamo due vincoli di disuguaglianza $g_1(x) \geq 0$, $g_2(x) \geq 0$ ed un punto $\overline{x}$, dove entrambi sono attivi.
Indichiamo con $\nabla g_1(\overline{x})$ e $\nabla g_2(\overline{x})$ la direzione della normale ai vincoli in $\overline{x}$.
Poiché i vincoli sono in forma di $\geq$, il gradiente punta verso l'interno della regione ammissibile.
Consideriamo un passo infinitesimo da $\overline{x}$ lungo una direzione d.
Per mantenere l'ammissibilità rispetto ai vincoli, d deve essere tale che:

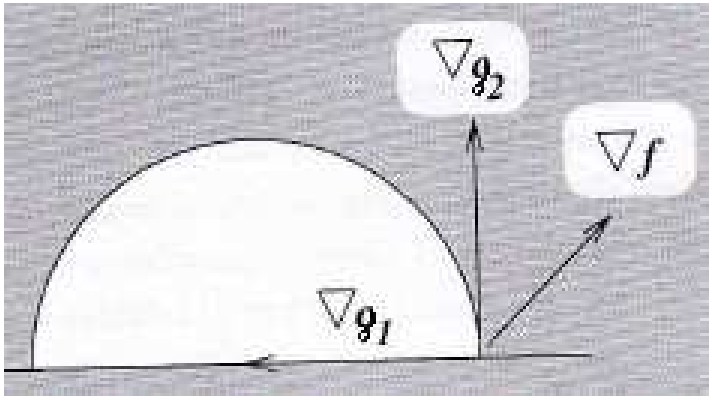
$$\nabla g_1(\overline{x})^T d \geq 0 \quad \text{e} \quad \nabla g_2(\overline{x})^T d \geq 0.$$

Il passo produce un miglioramento di $f(x)$ se e solo se

$$\nabla f(\overline{x})^T d < 0.$$

Vincoli di disuguaglianza





Direzioni ammissibili

Una direzione d è ammissibile in $\overline{x}$ se e solo se:

$$\nabla h_i(\overline{x})^T d = 0 \ \ \forall i \in \mathcal{E} \quad \text{e} \quad \nabla g_i(\overline{x})^T d \geq 0 \ \ \forall i \in \mathcal{A}(\overline{x}),$$

dove $\mathcal{A}(\overline{x})$ indica l'insieme dei vincoli di disuguaglianza attivi in $\overline{x}$.

Dobbiamo considerare solo i gradienti $\nabla g_i(\overline{x})$ linearmente indipendenti.
Per definire le direzioni in modo univoco, normalizziamo d in modo che abbia norma unitaria.

Un algoritmo delle direzioni ammissibili è un algoritmo iterativo che seleziona una direzione ammissibile ad ogni iterazione e poi calcola un passo ottimale lungo di essa risolvendo un problema non-lineare a singola variabile *vincolato*.