

# 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 Euclidea,
- eccetera...

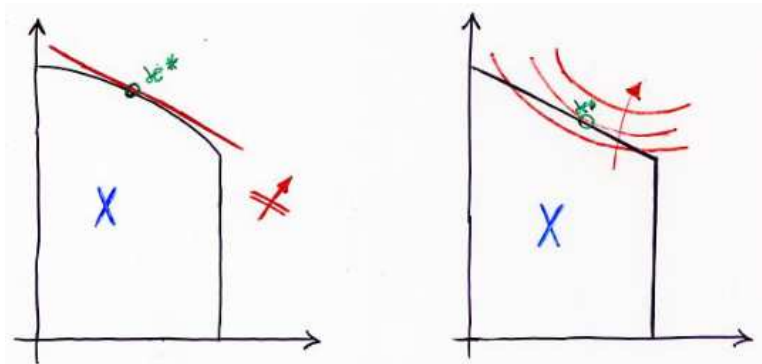
## Programmazione non-lineare (PNL)

Forma generale:

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

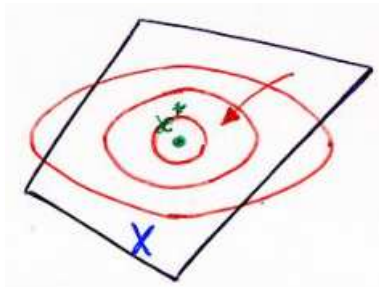
dove  $f(x)$ ,  $g(x)$  e  $h(x)$  possono essere funzioni non-lineari.

## Programmazione non-lineare (PNL)



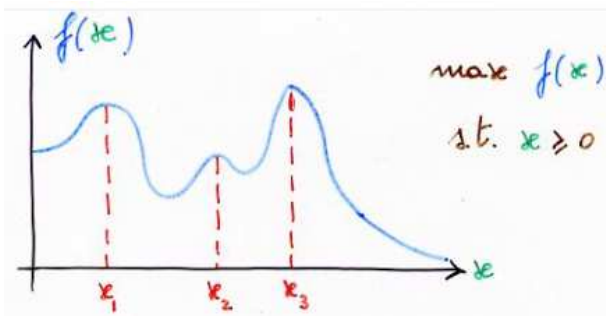
In generale, la soluzione ottima può non essere all'intersezione dei vincoli.

## 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  $\bar{x} \in X$  è un **minimo locale** se e solo se

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

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

## 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

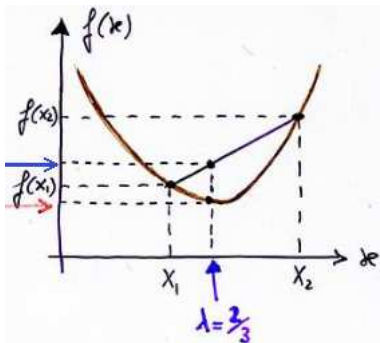
- la funzione-obiettivo è una **funzione convessa**;
- la regione ammissibile è un **insieme convesso**.



## 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  $f(x_1, x_2, \dots, x_n)$  è il vettore delle sue derivate parziali di **primo ordine**:

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

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

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

## Caratterizzazione dei minimi locali

**Condizioni necessarie del primo ordine.**

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

**Condizioni necessarie del secondo ordine.**

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

**Condizioni sufficienti del secondo ordine.**

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

## 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:

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

## Velocità di convergenza

Sia  $\{x_k\}$  una sequenza in  $\mathfrak{R}^n$  che converge a  $x^*$ .

Convergenza lineare:

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = r < 1.$$

Convergenza superlineare:

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

Convergenza quadratica:

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} = M.$$



## Algoritmi *line search*

Le due principali strategie sono:

- line search;
- trust regions.

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

## Metodo del gradiente

Per il teorema di Taylor

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

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

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

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

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

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

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

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

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

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

Metodi *quasi-Newton*, basati sull'approssimazione dell'Hessiano, sono stati ideati per ovviare a questo limite.

## Metodi trust region

I metodi *trust region* richiedono di

- approssimare la funzione  $f()$  con un modello  $m_k()$ , che viene aggiornato ad ogni iterazione  $k$ ;
- cercare un minimo di  $m_k()$  in un intorno di raggio  $\rho_k$  della soluzione corrente  $x_k$ .

Se la diminuzione del valore dell'obiettivo non è "grande abbastanza", il raggio viene diminuito e l'ottimizzazione viene ripetuta.

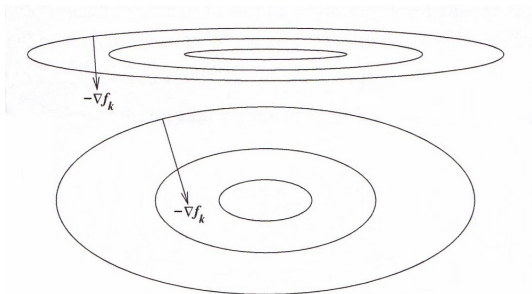
Di solito il modello  $m$  è quadratico

$$m_k(x_k + p) = f_k + p^T \nabla f_k + \frac{1}{2} p^T B_k p$$

e usa il gradiente  $\nabla f_k$  e l'Hessiano o una sua approssimazione  $B_k$ .

## Scaling

Gli algoritmi di programmazione non-lineare possono essere più o meno robusti rispetto allo **scaling**, che si ha, ad esempio, cambiando l'unità di misura di alcune grandezze.



L'**invarianza di scala** è una proprietà desiderabile degli algoritmi, che li rende più **robusti**.