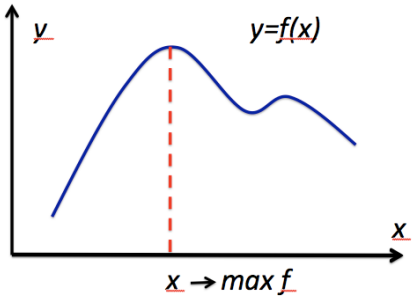
Otimering

- Med optimering menar vi maximering eller minimering av en funktion där de oberoende variablerna i vissa fall uppfyller bivillkor
- Typiska tillämpningar: inom fysik - minimera energin, inom ekonomi – minimera kostnad, maximera inkomst
- Vi börjar utan bivillkor: Finn *x* som maximerar (eller minimerar) *f*(*x*)



En dimension - sökning

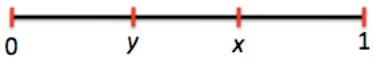
- Sökalgoritm, jämför halveringsmetoden för att lösa skalära ekvationer
- Antag att funktionen är unimodal, dvs den har endast ett max (eller min)
- Vi önskar sökalgoritm som garanterar att maximum ligger i visst interval och intervallets längd iterativt kan minskas
- Låt intervallet till en början vara (0,1) med 0 < *x* < 1, så att

$$f(x) > \max(f(0), f(1))$$

- Beräkna *f*(*y*), *y* < *x*. Intervallet med max kan nu minskas och algoritmen upprepas därefter i det mindre intervallet

$$\max i\,(0,x)\ om\ f(x) < f(y)$$
$$\max i\,(y,1)\ om\ f(x) > f(y)$$

En dimension - sökning



- För att fortsätta symmetrisk bör *x* dela (0,1) med gyllene snittet

$$\frac{x}{1} = \frac{y}{x} = \frac{1-x}{x} \rightarrow x^2 = 1 - x \rightarrow x = \frac{\sqrt{5}-1}{2} \approx 0.62$$

En dimension - Newton

- Derivatan är noll vid max (och min) för kontinuerligt deriverbara funktioner

$$f(x) \ \max vid \ x = \xi \rightarrow f'(\xi) = 0$$

- Använd nu Newtons algoritm på liknande sätt som vid ekvationslösning

$$\begin{cases} x_0 \ \ startvärde \\ x_{j+1} = x_j - \dfrac{f'(x_j)}{f''(x_j)}, \quad j = 0,1,.. \end{cases}$$

- Viktigt att välja bra startvärden

Flera dimensioner - Newton

$$f(x) \ \max \ vid \ x = \xi \rightarrow \nabla f(\xi) = 0$$

$$\begin{cases} x_0 \ \ startvärde \\ x_{j+1} = x_j - H(x_j)^{-1} \nabla f(x_j), \quad j = 0,1,.. \end{cases}$$

$$H(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)$$

- Hessian matrisen *H* kan beräknas med numeriska dividerade differenser om andraderivatorna är okända
- Det finns också algoritmer som liksom sekantmetoden ej behöver högre detivator

Flera dimensioner – gradient metoden

- Gradientmetoden är mest känd som "steepest descent" för minimering. (Vanligare än "steepest acsent" för maximering)

$$\begin{cases} x_0 \ \ startvärde \\ x_{j+1} = x_j + \alpha_j \nabla f(x_j), \quad j = 0,1,.. \end{cases}$$

$$\alpha_j \ \ beräknas \ för \ att \ maximera \ \ f(x_j + \alpha_j \nabla f(x_j))$$

$$(en - dim. maximering)$$

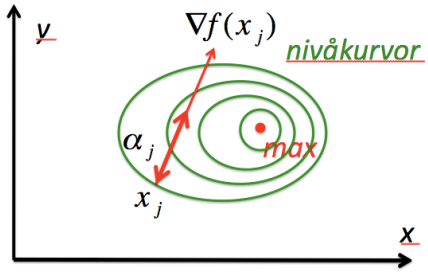
- Sökmetoder är inte använda i flera dimensioner förutom som endimensionellt delproblem i ovanstående algoritm

Gradient metoden

$$x_{j+1} = x_j + \alpha_j \nabla f(x_j)$$

α_j *beräknas för att maximera*

$$f(x_j + \alpha_j \nabla f(x_j))$$



Problem med bivillkor

- Två typer av metoder är vanliga.
  (1) transformera problemet till problem utan bivillkor
  (2) utvekla algoritm som inkluderar bivillkoren

(1) Straffunktionsteknik. Addera straffunktion till $f(x)$ så att bivillkoren uppfylls automatiskt vid maximering.

$$\max_{x \in D} f(x) \rightarrow \max_{x} \left(f(x) + g_\varepsilon(x) \right)$$

$$g_\varepsilon(x) \rightarrow -\infty, \ \textit{då} \ \ x \rightarrow \partial D, x \in D$$

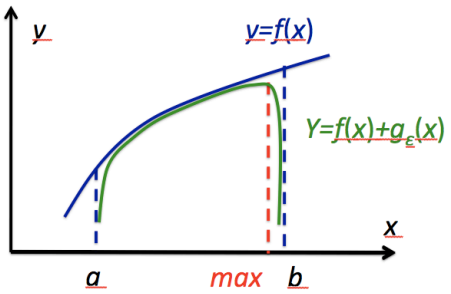
$$\partial D: \ \textit{randen av } D$$

Exempel i 1D

$$\max_{a \le x \le b} f(x) \rightarrow \max_{x} \left(f(x) + g_\varepsilon(x) \right)$$

$$g_\varepsilon(x) = \frac{\varepsilon}{(x-a)(x-b)}$$

- Starta med x_0 i (a,b) och finn x som ger max
- Använd detta x-värde som start för ny maximering med mindre ε



Algoritm med bivillkor – linjär programmering

- Linjär *f* och linjära bivillkor

$$\max_{x \in D \subset R^d} f(x) \rightarrow \max_{x \in D} \sum_{j=1}^{d} a_j x_j$$

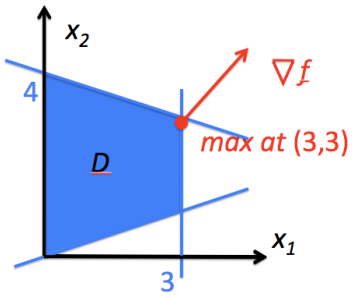
$$D: \sum_{j=1}^{d} b_{k,j} x_j \le c_k, \quad k = 1,2,..$$

- Maximum antas för *x* i ett hörn av *D*

Exempel, *d*=2

$$\max_{x \in D}(x_1 + x_2)$$

$$D: \begin{cases} x_1 + 3x_2 - 12 \le 0 \\ x_1 - 3x_2 \le 0 \\ x_1 - 3 \le 0 \\ -x_1 \le 0 \end{cases}$$



- Simplex metoden ger en systematisk väg att gå från hörn till hörn till maximum nås. Metoden fungerar i flera dimensioner

Matlab

- Matlab funktionerna max och min bestämmer extermvärden i en matris
- Funktionen fminbnd(fun,x1,x2) finner x-värdet för min av funktionen fun i intervallet (x1,x2) (För att bestämma max, minimera –fun)
- Matlabs toolbox optimization innehåller flera rutiner för max och min med eller utan bivillkor