Optimointi

Optimointi on funktion minimiarvon etsimistä.

Haettava funktion $f(\mathbf{x})$ minimi, $\mathbf{x} = (x_1, \ldots, x_N)$. Funktio f on kohdefunktio (objective function).

Jos tehtävään ei liity mitään lisäehtoja, kyseessä on rajoitteeton optimointitehtävä.

Rajoitetussa tehtävässä ratkaisu on lisäksi toteutettava annetut rajoitusehdot, jotka voivat olla
- epäyhtälörajoitteita: $g(\mathbf{x}) \leq 0$
- yhtälörajoitteita: $h(\mathbf{x}) = 0$.

Usean muuttujan derivaattoja käyttävissä menetelmissä tarvitaan toisista derivaatoista muodostuvaa Hessen matriisia

$$H(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{pmatrix}.$$

Hyvin monet tehtävät voidaan muuntaa optimointitehtäviksi:

1 Yhtälön $f(x) = 0$ ratkaisun etsiminen. Nollakohta on myös funktion $|f|$ minimi. Koska kyseessä on alhaalta rajoitettu funktio, sillä on aina ainakin infimum, vaikka alkuperäisellä yhtälöllä ei olisi ratkaisua. Erityisesti epälineaaristen yhtälöiden ryhmä on usein vaikeasti ratkaistavissa, jolloin sitä voi yrittää ratkaista optimointitehtävänä.

2 Pienimmän neliösumman sovitus tapahtuu tavallisesti minimoimalla residuaalia. Epälineaarisissa sovituksissa analyyttista ratkaisua ei yleenä löydy, joten on etsittävä suoraan residuaalin R minimikohtaa. Sovitettavan funktion muodolla ei ole oleellista vaikutusta tehtävän mutkikkuuteen.

3 Jos sovituksen kriteerinä on jokin muu kuin pienin neliösumma, residuaalin optimointi on usein ainoa ratkaisumenetelmä.

4 Regularisoitujen tehtävien ratkaiseminen. Regularisointiparametri voi olla yksi etsittävistä muuttujista.

5 Kombinatoriset ongelmat (esim. kauppamatkustajan ongelma). Jotkin mahdollisesti NP-täydellisiä; ratkaisuun tarvittava aika luokkaa 2^n tai $n!$.

Lokaali optimointi

1. Funktion derivaatat tunnetaan Esim. liittogradienttimenetelmät.

2. Derivaattoja ei tunneta tai ne ovat hankalia laskea. Esim. polytooppimenetelmä.

Globaali optimointi

Täysin luotettavia algoritmeja ei ole.

- Haetaan suuri määrä lokaaleja optimeja aloittamalla eri lähtöpisteistä.
- Geneettiset algoritmit.
- Simuloitu jäähdytys.

Haarukointimenetelmä

Yksiulotteinen tapaus; derivaattaa ei tunneta.

Tunnetaan funktion arvo kolmessa pisteessä $a < b < c$, s.e. $f(b) < f(a)$ ja $f(b) < f(c)$.

Valitaan piste x esimerkiksi väliltä (b, c). Jos $f(x) < f(b)$, uusi väli on (b, c), muuten (a, x).

Toistetaan sama toimenpide, kunnes $f(x)$ ei enää pienene.

Minimin kohdalla $f'(x) = 0$, joten f:n muutos on korkeintaan toista kertalukua. Iterointi voidaan lopettaa, kun välin pituus on luokkaa $\sqrt{\epsilon}$, missä ϵ on konevakio. Tätä pienemmät x:n muutokset eivät enää näy funktion arvoissa.

Kultaisen leikkauksen sääntö: välin keskimmäinen piste valitaan aina niin, että

$$\frac{b-a}{c-a} = \frac{3-\sqrt{5}}{2} = 0.38197.$$

Seuraava piste valitaan pitemmältä osaväliltä s.e.

$$\frac{x-b}{c-b} = 0.38197.$$

Yksiulotteinen tapaus; derivaatta tunnetaan

Oletetaan taas, että $a < b < c$ ja $f(b) < f(a)$ ja $f(b) < f(c)$.

Lasketaan $f'(b)$. Jos $f'(b) > 0$, seuraava piste valitaan väliltä (a, b), muuten väliltä (b, c).

Kun on laskettu derivaatta kahdessa pisteessä, voidaan esim. sekanttimenetelmällä laskea seuraava piste, jossa derivaatan pitäisi olla nolla. Jos näin laskettu piste on välin ulkopuolella, otetaan uudeksi pisteeksi välin keskipiste.

N-ulotteinen tapaus

Edetään johonkin suuntaan niin kauan kuin kohdefunktion arvo pienenee, jonka jälkeen muutetaan suuntaa. Tätä viivahakua toistetaan, kunnes ei enää löydy pienempiä arvoja.

Kullakin askeleella ratkaistaan yksiulotteinen minimointitehtävä. Eteneminen tiettyyn suuntaan päättyy, kun kohdefunktion gradientilla ei ole etenemissuunnan suuntaista komponenttia.

Etenemissuunta voidaan valita eri tavoin.

1. Edetään vain koordinaattiakselien suuntaan; muutetaan vuorotellen koordinaatteja $x_1, \ldots, x_N, x_1, \ldots$. Usein tehotonta.

2. Edetään jyrkimmän muutoksen (eli kohdefunktion gradientin) suuntaan. Kunkin viivahaun jälkeen käännytään aina 90°. Ei välttämättä johda kovin suoraan kohti minimiä.

3. Yritetään valita seuraava suunta siten, että otetaan huomioon aikaisemmat suunnat.

Liitto- eli konjugaattigradienttimenetelmä

Olkoon $\mathbf{x}$ pisteen paikka toistaiseksi parhaan minimikohdan $\mathbf{p}$ suhteen. Silloin on

$$\begin{aligned} f(\mathbf{x}) &\approx f(\mathbf{p}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j \\ &= c + \mathbf{b} \cdot \mathbf{x} + \frac{1}{2} \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}, \end{aligned}$$

missä $\mathbf{b}$ on funktion f gradientti pisteessä $\mathbf{p}$ ja $\mathbf{A}$ Hessen matriisi pisteessä $\mathbf{p}$. Tästä derivoimalla saadaan gradientille lauseke (pisteessä $\mathbf{x}$)

$$\nabla f = \mathbf{A} \cdot \mathbf{x} + \mathbf{b}.$$

Kun pisteestä $\mathbf{x}$ lähdetään johonkin suuntaan $\mathbf{v}$, gradientin muutos on $\mathbf{A} \cdot \mathbf{v}$.

Oletetaan, että edellinen askel on kuljettu suuntaan $\mathbf{u}$. Jotta seuraava askel ei heikentäisi tästä saatua hyötyä, gradientin täytyy olla kohtisuorassa vektoria $\mathbf{u}$ vastaan myös askelen $\mathbf{v}$ jälkeen:

$$\mathbf{u} \cdot \mathbf{A} \cdot \mathbf{v} = 0.$$

Tällaisia suuntia $\mathbf{u}$ ja $\mathbf{v}$ sanotaan toistensa konjugaateiksi.

1. Valitaan jokin alkuarvo $\mathbf{x}_1$.

2. Toistetaan seuraavaa, kun $k = 1, 2, \ldots$, kunnes $\| \mathbf{g}_k \|$ on riittävän pieni.

   - $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$.
   - jos $k = 1$, asetetaan $\beta_1 = 0$, $\mathbf{s}_0 = 0$; muuten

     $$\beta_k = \frac{\mathbf{g}_k \cdot \mathbf{g}_k}{\mathbf{g}_{k-1} \cdot \mathbf{g}_{k-1}}$$

   - $\mathbf{s}_k = -\mathbf{g}_k + \beta_k \mathbf{s}_{k-1}$.
   - etsitään viivahaulla piste $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{s}_k$. Edetään siis suuntaan $\mathbf{s}_k$, kunnes tullaan minimiin.

Polytooppimenetelmä

Nelderin ja Meadin polytooppimenetelmä; myös simpleksimenetelmä (eri asia kuin lineaaristen opitmointitehtävien simpleksimenetelmä). Kohdefunktion derivaattoja ei tarvita.

N-ulotteisen avaruuden simpleksi on monitahokas, jossa on $N + 1$ kärkeä $\mathbf{x}_1, \ldots, \mathbf{x}_{N+1}$. Merkitään $f_i = f(\mathbf{x}_i)$, $f_l = \min f_i$, $f_h = \max f_i$.

Huonointa kärkeä vastaavan vastakkaisen tahkon painopiste

$$\mathbf{x}_c = \frac{1}{n} \sum_{i \neq h} \mathbf{x}_i.$$

Peilauskerroin $\alpha = 1$, pienennyskerroin $\beta = 0.5$, laajennuskerroin $\gamma = 2$.

Simpleksiä muunnellaan neljänlaisilla operaatioilla:

1. Peilataan huonoin kärki vastakkaisen tahkon painopisteen suhteen:

$$\mathbf{x}_r = (1 + \alpha)\mathbf{x}_c - \alpha \mathbf{x}_h.$$

2. Laajennetaan simpleksia:

$$\mathbf{x}_e = (1 - \gamma)\mathbf{x}_c + \gamma \mathbf{x}_r.$$

3. Pienennetään simpleksiä siirtämällä yhtä kärkeä.

$$\mathbf{x}_s = (1 - \beta)\mathbf{x}_c + \gamma \mathbf{x}_h.$$

4. Pienennetään simpleksiä siirtämällä yhtä tahkoa.

```
program amoeba
! Nelder-Mead polytooppi- eli simpleksimenetelma
! muunneltu Numerical Recipes -kirjasta
 implicit none
 integer, parameter :: NMAX = 1000, &  ! askelten maksimimaara
                ndim = 2, &    ! avaruuden dimensio
                mpts = 3      ! simpleksin karkien maara
 real, parameter :: FTOL = 0.001, &   ! lopetustarkkuus
             ALPHA = 1.0, &    ! peilauskerroin
             BETA = 0.5, &     ! pienennyskerroin
             GAMMA = 2.0       ! venytyskerroin

 integer i,j,ilo,ihi,inhi, nfunk
 real ytry, ysave, sum, rtol, psum(ndim), &
    p(ndim+1, ndim), &
    y(ndim+1)

 ! ensimmaisen simpleksin karjet
 p(1,:) = (/ 0.0, 0.0 /)
 p(2,:) = (/ 0.0, 1.0 /)
 p(3,:) = (/ 1.0, 0.0 /)
 ! funktion arvot simpleksin karjissa
 do i=1,3
  y(i) = funk(p(i,:))
 end do
 nfunk=0

 ! koordinaattien summat; karjen i vastakkaisen
 ! simpleksin painopiste on (psum-p(i))/ndim
 do j=1,ndim
  sum=0.0
  do i=1,mpts
   sum = sum+p(i,j)
  end do
  psum(j) = sum
 end do

 main: do
  ! haetaan paras (ilo), huonoin (ihi) ja
  ! toiseksi huonoin (inhi) karkipiste
  ilo=1
  if (y(1) > y(2)) then
    inhi=2; ihi=1
  else
    inhi=1; ihi=2
  end if
  do i=1,mpts
   if (y(i) < y(ilo)) ilo=i
   if (y(i) > y(ihi)) then
     inhi=ihi; ihi=i
   else if (y(i) > y(inhi)) then
     if (i /= ihi) inhi=i
   end if
  end do
```

```
   ! jos huonoin ja paras arvo lahes samoja, lopetetaan
   rtol=2.0*abs(y(ihi)-y(ilo)) / (abs(y(ihi))+abs(y(ilo)))
   if (rtol < FTOL) then
        write(6,'("rtol=",f10.5)') rtol
        write(6,'("nfunk=",i5)') nfunk
        exit main
   end if
   ! jos iteraatiokierroksia liikaa, lopetetaan
   if (nfunk >= NMAX) then
        write(6,'("nfunk=",i5)') nfunk
        exit main
   end if


   ! peilaus
   ytry=amotry(-ALPHA)

   ! jos tulos parani, siirretaan uutta karkea viela kauemmas
   if (ytry <= y(ilo)) then
    ytry=amotry(GAMMA)
   else if (ytry >= y(inhi)) then
    ! uusi piste huonompi kuin toiseksi huonoin;
    ! pienennetaan simpleksia
    ysave=y(ihi)
    ytry=amotry(BETA)
    if (ytry >= ysave) then
     ! edelleen liian iso; pienennetaan simpleksia
     ! parhaan pisteen suhteen
     do i=1,mpts
      if (i /= ilo) then
       do j=1,ndim
        psum(j)=0.5*(p(i,j)+p(ilo,j))
        p(i,j)=psum(j)
       end do
       y(i)=funk(psum)
      end if
     end do
     nfunk = nfunk+ndim
     ! paivitetaan koordinaattien summat
     do j=1,ndim
      sum=0.0
      do i=1,mpts
       sum = sum+p(i,j)
      end do
      psum(j) = sum
     end do
    end if
   end if
  end do main

  ! tulostetaan karkipisteet ja funktion arvot; kaikkien
pitaisi
  ! olla melkein samoja, silla ratkaisun loytyessa simpleksi
on
  ! surkastunut optimipisteeksi
```

```
 write(*,*) p(1,:), y(1)
 write(*,*) p(2,:), y(2)
 write(*,*) p(3,:), y(3)

 contains

 ! optimoitava kohdefunktio
 real function funk(p)
  implicit none
  real p(ndim)
  funk=(p(1)-3.0)**2+(p(2)-2.0)**2+1.0
 end function


 ! simpleksin muunnosaskel
 ! huonoin karki peilataan vastakkaisen sivun suhteen ja
 ! etaisyytta muutetaan kertoimella fac
 real function amotry(fac)
  implicit none
  real fac, fac2
  integer j
  real fac1, ytry, ptry(ndim)
  fac1=(1.0-fac)/ndim
  fac2=fac-(1-fac)/ndim
  do j=1,ndim
    ptry(j)=psum(j)*fac1+p(ihi,j)*fac2
  end do
  ytry=funk(ptry)
  nfunk = nfunk+1
  if (ytry < y(ihi)) then
   y(ihi)=ytry
   do j=1,ndim
    psum(j) = psum(j)+ptry(j)-p(ihi,j)
    p(ihi,j)=ptry(j)
   end do
  end if
  amotry=ytry
 end function
end program
```

Rajoitteet

Ratkaisuvektoria päivitettävä niin, että rajoitteet pysyvät aina voimassa.

Rajoitteellista tehtävää voidaan käsitellä rajoittamattomien tehtävien ohjelmilla, jos rajoitteet korvataan sakko- ja estefunktioilla.

Lisätään optimoitavaan funktioon termi, joka on hyvin suuri sallitun alueen ulkopuolella. Jos esimerkiksi on optimoitava f rajoitteella $g(x) \leq 0$, voidaan kohdefunktioksi ottaa

$$f(x) + s \max\{0, g(x)\},$$

missä vakio s on sakkoparametri.

Estefunktio on funktio, joka kasvaa rajatta lähestyttäessä sallitun alueen reunaa.

Globaali optimointi

Funktiolla voi olla useita lokaaleja minimejä. Minimi voi olla hyvin kapea.

Mikään menetelmä ei takaa globaalin minimin löytymistä.

Valitaan riittävän suuri määrä alkuarvoja, joista lähtien suoritetaan lokaali optimointi. Toivotaan, että jokin löytyneistä lokaaleista minimeistä on myös globaali.

Varsinkin työläissä kombinatorisissa tehtävissä geneettiset algoritmit ja simuloitu jäähdytys saattavat olla käyttökelpoisia.

Geneettiset algoritmit

Jäljittelevät evoluutioprosessia.

- Muodostetaan joukko ratkaisuvektoreita, alkupopulaatio.
- Risteytetään populaation alkioita keskenään ja aiheutetaan satunnaisia mutaatioita.
- Uudeksi populaatioksi valitaan parhaat ratkaisut.
- Toistetaan, kunnes tulos ei enää parane.

Mutaatioiden vuoksi ratkaisu ei juutu ensimmäiseen löytyneeseen lokaaliin minimiin.

Simuloitu jäähdytys

(Oikeastaan mellotus, simulated annealing) Metropolis (1953).

Analogia: kun nestettä jäähdytetään riittävän hitaasti, se jäätyy säännölliseksi kidehilaksi, jonka energia on minimissään.

Kohdefunktio vastaa systeemin energiaa. Konfiguraatiota muutellaan satunnaisesti. Uusi konfiguraatio valitaan lämpötilasta riippuvalla todennäköisyydellä.

Kun lämpötila on korkea, hyväksytään kohtalaisella todennäköisyydellä myös siirtymiset huonompiin (korkeamman energian) tiloihin. Tällä pyritään estämään juuttuminen lokaaleihin minimeihin.

Kun lämpötila laskee, todennäköisyys hyväksyä huonompi konfiguraatio pienenee.

Sopivia sovelluksia esim. kombinatoriset ongelmat, joille ei tunneta nopeita ratkaisumenetelmiä.

Löytää yleensä melko hyvän ratkaisun, mutta ei takaa sen optimaalisuutta.