Numeerista lineaarialgebraa Python-kielellä

Antti Rasila
Matematiikan ja tilastotieteen laitos, Helsingin yliopisto

Johdanto

Tämä kirjoitus on jatkoa kahdelle aikaisemmalle Solmussa ilmestyneelle kirjoitukselleni matemaattisesta ohjelmoinnista Python-kielellä. Tässä kirjoituksessa käsitellään menetelmiä ja ongelmia, jotka esiintyvät suunnilleen ensimmäisen vuoden yliopistokursseilla. Ohjelmoinnin perusteita ei käsitellä, niiden osalta viittaan aikaisempiin kirjoituksiin. Kuten sarjan edellisessäkin artikkelissa, tässä kirjoituksessa pyritään lähestymään matemaattisia ongelmia ohjelmoijan näkökulmasta. Taustalla on myös ajatus, että ohjelmointitaitoja voi käyttää hyväksi matemaattisten ideoiden omaksumisessa.

Tässä osassa tutkittavat ongelmat liittyvät lineaarialgebran perusteisiin. Tällä nimellä aihetta ei koulukursseissa käsitellä, mutta aihe sivuaa vektorilaskentaa ja ensimmäisen asteen yhtälöryhmien ratkaisemista. Lineaarialgebralla on paljon käytännöllisiä ja teoreettisia sovelluksia mm. tilastotieteeseen, teknillisiin tieteisiin ja useille matematiikan erityisaloille. Yliopistossa opiskelleet muistavat opiskeluajoiltaan lineaarialgebran peruskurssin työläänä laskemispainotteisena kurssina, jossa tehtäviä on paljon ja teoriaa suhteellisen vähän. Tässä kirjoituksessa esitetään vain keskeiset ideat ja annetaan koneen hoitaa laskeminen. On hyvä kuitenkin muistaa, että koneet eivät pysty korvaamaan asioiden ymmärtämistä. Asioiden ymmärtäminen puolestaan vaatii aina harjoittelua, joka voi olla esimerkiksi käsin laskemista tai tietokoneohjelmien kirjoittamista.

Numerical Python -laajennus tuo Python-kieleen mukaan laskennallisesti tehokkaita numeeriseen lineaarialgebraan liittyviä algoritmeja. Numerical Python on vapaa ohjelmisto ja sen voi ladata ilmaiseksi www-sivulta `http://sourceforge.net/projects/numpy`. Helposti asennettava Python-paketti, jossa myös Scientific Python ja Numerical Python -laajennukset ovat valmiiksi mukana, on ladattavissa ilmaiseksi www-sivulta `http://www.enthought.com/`.

Tehokkuuden parantamiseen käytetään tekniikkaa, jota kutsutaan vektoroinniksi. Vektoroinnin perusideana on, että samankaltainen operaatio halutaan suorittaa samalla kertaa suurelle määrälle alkeistapauksia. Nämä alkeistapaukset esitetään yksi- tai usempiulotteisena taulukkona. Vektorointi parantaa ohjelman suoritusaikoja kahdella tavalla:

1. Ohjelmointikielen tulkkia ei tarvitse kutsua eri operaatioiden välillä, eli tulkin aiheuttama hitaus jakautuu useille suoritettaville operaatioille. Tätä ideaa käytettävät esimerkiksi Numerical Python ja MATLAB.

2. Datavektorin eri alkioiden välillä ei ole riippuvuuksia, joten vektoroidut laskutoimitukset voidaan suorittaa rinnakkain. Tämä voidaan toteuttaa joko nopeana sarjatyöskentelynä (nk. vektoriprosessorit) tai jakamalla tehtävä useille eri prosessoreille. Nämä tekniikat ovat yleisiä supertietokoneissa ja leviämässä myös kotitietokoneisiin (esim. SSE2 ja Altivec).

Vektorointi tarjoaa mm. numeeriseen lineaarialgebraan sopivia toimintoja, mutta laskennassa esiintyviä vektoreita ei pidä sekottaa matemaattisiin vektoreihin. Niillä tehtävät laskutoimitukset tarkoittavat elementeittäin suoritettavia skalaarien laskutoimituksia. Ohjelmoinnin helpottamiseksi vektoreille voi yleensä suorittaa myös joitakin muita operaatioita, joista tavallisimpia ovat maksimi, minimi ja elementtien summa.

Esimerkkejä. Oletetaan, että $n \geq 1$ ja $\bar{x} = [x_1, x_2, x_3, \ldots, x_n]$ ja $\bar{y} = [y_1, y_2, y_3, \ldots, y_n]$ ovat n:n mittaisia vektoreja (Python-mielessä). Tällöin

$$\begin{aligned} \bar{x}^2 &= [x_1^2, x_2^2, x_3^2, \ldots, x_n^2], \\ \sin(\bar{x}) &= [\sin(x_1), \sin(x_2), \sin(x_3), \ldots, \sin(x_n)], \\ \bar{x} * \bar{y} &= [x_1 y_1, x_2 y_2, x_3 y_3, \ldots, x_n y_n], \\ \bar{x} + \bar{y} &= [x_1 + y_1, x_2 + y_2, x_3 + y_3, \ldots, x_n + y_n]. \end{aligned}$$

Huomaa, että ylläolevista laskutoimituksista ainoastaan $\bar{x} + \bar{y}$ on järkevä myös matemaattisten vektorien laskutoimituksena.

Numerical Python -kirjasto

Numerical Pythonin (Numeric) tarkoituksena on tarjota kaupallista MATLAB-ohjelmistoa vastaava vapaa avoimen lähdekoodin ohjelmointiympäristö numeerisen matematiikan sovelluksille. Erityisesti Numerical Python -kirjastossa on toteutettu nopeat vektoroidut laskutoimitukset, jotka pienentävät käännetyllä ohjelmointikiellä (esimerkiksi C, C++ tai Fortran) saatavaa tehokkuusetua Pythoniin nähden. Vektoroitujen skalaarilaskutoimitusten lisäksi kirjastosta löytyy perusalgoritmeja lineaarialgebraan, jotka on sisällytetty erilliseen *LinearAlgebra*-pakettiin. Vaikka Numerical Python onkin melko monipuolinen ohjelmointikirjasto, kaikkia MATLAB-kielen ominaisuuksia ei siinä ole toteutettu, esimerkiksi kuvien piirtoon ja käsittelyyn ei tarjota mitään. Puuttuvia piirteitä on toteutettu mm. *Scientific Python*-laajennuksessa sekä visualisointikirjastossa *MatPlotLib*. Suhteellisen hyvän tehokkuuden ja vapaan levitettävyyden lisäksi tämän ympäristön tekee opetuskäytössä houkuttelevaksi Python-ohjelmointikielen sopivuus ensimmäiseksi ohjelmointikieleksi ohjelmoinnin alkeisopetuksessa.

Seuraavassa esimerkkejä vektoroiduista sklaarilaskutoimituksista Numerical Pythonia käyttäen. Risuaidalla merkityt rivit ovat kommentteja, eikä niitä tarvitse kirjoittaa.

```
# ladataan Numerical Python -kirjasto
>>> from Numeric import *
# käsky arange(a,b) tuottaa vektoriin
>>> x=arange(1.0,5.0)
# luvut a:sta b:hen
# vektoreita voidaan alustaa myös näin
>>> y=array([0,1,0,1])
>>> x                       # tulostetaan x
array([ 1.,  2.,  3.,  4.])
>>> y
array([0, 1, 0, 1])
>>> x*y                     # x:n ja y:n tulo
array([ 0.,  2.,  0.,  4.])
>>> x**y # vektoroitu poteenssinkorotus
array([ 1.,  2.,  1.,  4.]) # x potenssiin y
# vektoroituja laskutoimituksia
>>> sin(x)
array([ 0.84147,  0.90929,  0.14112, -0.75680])
>>> sin(x)+cos(x)
array([ 1.38177,  0.49315, -0.84887, -1.41044])
```

Matriisit ja lineaariset yhtälöryhmät

Tässä luvussa esitetään lyhyt johdatus matriiseihin ja lineaarisiin yhtälöryhmiin. Tarkastellaan aluksi seuraavaa yhtälöryhmää:

$$\text{(1)} \qquad \begin{cases} a_{11}x_1 + a_{12}x_2 &= b_1, \\ a_{21}x_1 + a_{22}x_2 &= b_2. \end{cases}$$

Tässä a_{ij}:t ja b_i:t ovat vakioita ja x_1, x_2 ovat muuttujia. Tällaista yhtälöryhmää kutsutaan lineaariseksi yhtälöryhmäksi.

Huomataan, että yhtälöryhmässä esiintyvät kertoimet a_{ij} määrittevät funktion eli kuvauksen, joka kuvaa vektorin (x_1, x_2) vektorille (b_1, b_2). Merkitään tätä kuvausta kirjaimella A ja huomataan, että $A\colon \mathbf{R}^2 \to \mathbf{R}^2$. Sovitaan, että jatkossa ylläolevaa yhtälöryhmää tarkoitettassa kirjoitetaan $A(x) = b$. Kun halutaan kertoa, että kertoimet a_{ij} määrittelevät kuvauksen A, kirjoitetaan

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

Tätä esitystä nimitetään 2×2-matriisiksi. Samalla tavoin voidaan määritellä $m \times n$-matriiseja. Jos $n = m$, kutsutaan matriisia neliömatriisiksi. Vektorit tulkitaan lineaarialgebrassa tavallisesti $n \times 1$-matriiseksi.

Jos A, B ovat $m \times n$-matriiseja,

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix},$$

$$B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix},$$

määritellään matriisien summa

$$A+B = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \dots & a_{mn}+b_{mn} \end{bmatrix}.$$

Jos A on $m \times p$-matriisi ja B on $p \times n$-matriisi, niin matriisien A ja B tulo määritellään kaavalla

$$A \cdot B = \begin{bmatrix} \sum_1^p a_{1k}b_{k1} & \sum_1^p a_{1k}b_{k2} & \dots & \sum_1^p a_{1k}b_{kn} \\ \sum_1^p a_{2k}b_{k1} & \sum_1^p a_{2k}b_{k2} & \dots & \sum_1^p a_{2k}b_{kn} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_1^p a_{mk}b_{k1} & \sum_1^p a_{mk}b_{k2} & \dots & \sum_1^p a_{mk}b_{kn} \end{bmatrix}.$$

Matriisin kertominen reaaliluvulla t määritellään $tA = (ta_{ij})$. Matriisien A ja B tulo AB vastaa yhdistettyä kuvausta $A \circ B$. Ylläolevan kaavan voi siis johtaa laskemalla lausekkeen yhdistetylle kuvaukselle. Lineaarinen yhtälöryhmä (1) voidaan kirjoittaa myös matriisitulon avulla muotoon $Ax = b$, missä x on 2×1-matriisi.

Neliömatriisin A käänteismatriisi A^{-1} on sellainen matriisi, että $AA^{-1} = I$. Tässä I on yksikkömatriisi

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix},$$

Käänteismatriisi A^{-1} vastaa A:n käänteiskuvausta (jos sellainen on olemassa), ja yksikkömatriisi vastaa identtistä kuvausta. Erityisesti, jos $Ax = y$ niin $A^{-1}y = x$. Matriisilla A on käänteismatriisi jos ja vain jos A:n determinantti $\det A \neq 0$. Jos A on 2×2-matriisi, niin

$$\det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

Lineaarisen yhtälöryhmän ratkaiseminen

Matriisin lineaaristen yhtälöryhmien käsittelemiseen Numerical Python tarjoaa kirjaston LinearAlgebra. Ohessa esimerkkejä kirjaston käytöstä matriiseilla laskemiseen.

```
>>> from Numeric import *
>>> from LinearAlgebra import *
>>> A=array([[1,2],[3,4]])
>>> A
array([[1, 2],
       [3, 4]])
>>> b=array([5,6])
>>> iA=inverse(A)
>>> iA
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
>>> matrixmultiply(A,iA)
array([[  1.00000000e-00,   1.11022302e-16],
       [ -2.22044605e-16,   1.00000000e+00]])
>>> matrixmultiply(iA,b)
array([-4. ,  4.5])
```

LinearAlgebra-kirjastosta löytyy valmiita algoritmeja lineaarisen yhtälöryhmän ratkaisemiseen ja käänteismatriisin laskemiseen. Seuraavaksi käydään läpi eräs tapa, joka on helppo ymmärtää, mutta ei laskennallisesti tehokas.

Kuten edellä 2×2-matriisin tapauksessa todettiin, yhtälöryhmä (1) voidaan kirjoittaa muotoon $Ax = b$. Samoin voidaan menetellä tapauksessa, jossa A on $n \times n$-matriisi. Seuraavaksi tarkastelemme tällaisen yleisen tapauksen tärkeää erikoistapausta, jossa matriisilla A on erityinen muoto, sen *diagonaalin* alapuoliset alkiot ovat nollia. Matriisia U sanotaan yläkolmiomatriisiksi, jos se on muotoa

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix},$$

missä kaikki diagonaalin alapuoliset alkiot ovat nollia[1]. Jos U on yläkolmiomatriisi, lineaarinen yhtälöryhmä $Ux = b$ voidaan ratkaista seuraavasti:

1. Huomataan, että itseasiassa alin yhtälö on jo melkein ratkaistu. Sen ratkaisu on $x_n = b_n/u_{nn}$.
2. Sijoitetaan saatu x_n:n arvo muihin yhtälöihin ja muuttuja x_i. Ratkaisu saadaan kaavasta

$$x_i = \frac{1}{u_{ii}}\Big[b_i - \sum_{j=i+1}^{n} u_{ij}x_j\Big], \text{ kun } k = n-1, \dots, 2, 1.$$

Seuraavaksi esitetään tapa, jolla lineaarinen yhtälöryhmä $Ax = b$ ($\det A \neq 0$) voidaan palauttaa sellaiseksi, jossa esiintyy yläkolmiomatriisi U, eli $Ux = \tilde{b}$. Tätä menettelyä kutsutaan Gaussin eliminoinniksi.

1. Lähdetään liikkeelle yhtälöryhmän ylimmältä riviltä. Kerrotaan ylin rivi luvulla $c_1 = -a_{21}/a_{11}$ ja lasketaan tulos yhteen toiseksi ylimmän kanssa, jolloin muuttujan x_1 kerroin häviää. Sijoitetaan lopputulos toiseksi ylimmän rivin paikalle. Jos $a_{11} = 0$, rivien järjestystä pitää vaihtaa.

[1] Vastaavasti matriisia, jossa diagonaalin yläpuoliset alkiot ovat nollia, sanotaan alakolmiomatriisiksi.

2. Toistetaan sama menettely vakiolla $c_2 = -a_{31}/a_{11}$ kolmannella rivillä, jolloin x_1:hden kerroin häviää myös kolmannelta riviltä.
3. Edelleen toistetaan menettely lopuilla riveillä, ja saadaan yhtälöryhmä, jossa x_1:hden kerroin on nolla kaikilla muilla riveillä paitsi ensimmäisellä.
4. Aloitetaan toiselta riviltä. Menetellen kuten edellä, hävitetään muuttujan x_2 kertoimet riveiltä 3-n.

Lopputuloksena siis on yhtälöryhmä, joka on muotoa $Ux = \tilde{b}$.

Gaussin eliminointi on työlästä laskea käsin, mutta tietokoneella se sujuu helposti. Lopputuloksena saatava yhtälöryhmä voidaan edelleen ratkaista kuten yllä. Käytännössä lineaarisen yhtälöryhmän ratkaiseminen kannattaa tehdä LinearAlgebra-paketin valmiilla algoritmeilla.

```
>>> solve_linear_equations(A,b)
array([-4. ,  4.5])
```

Muita paketistä löytyviä hyödyllisiä käskyjä ovat mm. matriisin a determinantti `determinant(a)` ja käänteismatriisi `inverse(a)`.

Linkkejä ja lisämateriaalia

- Python-ohjelmointikielen kotisivu (Pythonin voi ladata täältä): `http://www.python.org/`
- Ohjelmoinnin peruskurssi Pythonilla (lukiotasoinen ohjelmointikurssi): `http://www.cs.helsinki.fi/u/vahakota/aott/index.html`
- Numerical Pythonin kotisivu: `http://sourceforge.net/projects/numpy`
- SciPy – Scientific tools for Python: `http://www.scipy.org/`
- Python Tutorial: `http://www.python.org/doc/current/tut/tut.html`
- Dive Into Python: Python from novice to pro (verkosta ladattava kirja): `http://www.diveintopython.org/`
- (Yliopistotaisoisen) kurssin "Lineaarialgebra ja matriisilaskenta" luentomoniste: `http://mathstat.helsinki.fi/kurssit/info/lineaarialgebra_ja_matriisilaskenta04s.html`
- Virtuaalista lineaarialgebran oppimateriaalia (mm. harjoitustehtäviä): `http://www.it.lut.fi/mat/virtuaali/matb/list.html?root=382`
- Professor Strang's Linear Algebra Lecture Videos (englanninkielinen videokurssi, MIT OpenCourseWare): `http://ocw.mit.edu/OcwWeb/Mathematics/18-06Linear-AlgebraFall2002/VideoLectures/index.htm`