Calcul Numeric

Laboratorul 1.

Regresia liniară

1 Introducere

Soluțiile problemelor matematice necesită adaptare pentru a putea fi implementate numeric. Pe de o parte, unele metode nu pot fi traduse întocmai în algoritmi generali de calcul (vezi de exemplu, calculul integralelor sau rezolvările ce presupun serii Taylor). Pe de altă parte, reprezentarea numerelor reale în format finit, cum este formatul virgulă mobilă, implică erori de rotunjire: un număr nu poate fi reprezentat oricât de precis, iar erorile de rotunjire depind de numărul de cifre semnificative ale reprezentării. Anumite calcule pot amplifica aceste erori, ducând la soluții numerice eronate, în ciuda corectitudinii lor matematice. Pentru a putea controla acest fenomen, este utilă noțiunea de epsilon-mașină (ϵ), ce măsoară distanța dintre reprezentarea numărului 1 și cea a numărului imediat următor. De notat faptul că în formatul virgulă mobilă numerele nu sunt echidistante, astfel că distanța dintre 1 și următorul număr nu este aceeași cu cea dintre 100 și următorul număr. Standardul IEEE impune valoarea $\epsilon = 2.2 \times 10^{-16}$. Prin urmare, erorile de rotunjire vor avea ordinul de mărime al lui ϵ.

1.1 Condiționare și stabilitate numerică

În continuare tratăm două noțiuni înrudite ce caracterizează problemele matematice și metodele de calcul ale acestora: condiționarea și stabilitatea numerică. Ambele se referă la efectul pe care schimbarea input-ului (datele problemei, valorile variabilelor) le are asupra output-ului (rezultatul). Condiționarea este o proprietate ce caracterizează problema matematică, în timp ce stabilitatea algoritmul de calcul.

Presupunem că putem exprima o problemă ca o funcție $\boldsymbol{y} = f(\boldsymbol{x})$. Numărul de condiționare al problemei este o mărime ce măsoară sensibilitatea la perturbații a variabilei $\boldsymbol{x}$. Cu alte cuvinte, dacă $\hat{\boldsymbol{x}}$ reprezintă o altă valoare a intrării, iar $\hat{\boldsymbol{y}} = f(\hat{\boldsymbol{x}})$, atunci numărul de condiționare $C(x)$ satisface relația

$$|\hat{\boldsymbol{y}} - \boldsymbol{y}| = C(x)|\hat{\boldsymbol{x}} - \boldsymbol{x}| \tag{1}$$

Cu cât valoarea este mai mare, cu atât problema este mai rău condiționată. Observați faptul că numărul de condiționare depinde de intrarea $\boldsymbol{x}$. O ilustrare a acestei proprietăți va fi prezentată în Secțiunea următoare.

Dacă un algoritm are o acuratețe de ordinul numărului de condiționare al problemei pe care o rezolvă, el se numește stabil numeric. Eliminarea Gaussiană, spre exemplu, nu este un algoritm stabil, de aceea sunt necesare strategiile de pivotare. Există însă și excepții, printre care matricile simetrice pozitiv definite și cele strict diagonal dominante, pentru care pivotarea nu este necesară.

1.2 Eficiența calculului matricial

Librării ca numpy utilizează rutine low-level de algebră vectorială pentru a executa în mod eficient operații vectoriale și matriciale. Dintre acestea, cele mai importante sunt BLAS (Basic Linear Algebra Subprograms) [1] și LAPACK (Linear Algebra PACKage) [2]. BLAS este organizat pe nivele, în care primul conține operații scalare, la nivel de vector și operații între doi vectori, nivelul 2 conține implementări pentru operații matrice-vector, iar ultimul nivel operații matrice-matrice. Acestea sunt utilizate și în rutinele LAPACK, care conțin soluții eficiente pentru rezolvarea sistemelor liniare, calculul valorilor proprii și al valorilor singulare ale unei matrici și alte probleme de algebră lineară.

Matricile sunt tablouri bidimensionale, în timp ce memoria calculatoarelor poate fi reprezentată de o structură de date de tip vector. Din acest motiv, pentru a putea fi stocate, e necesar ca matricile să fie vectorizate. Această operație se poate face fie prin alipirea liniilor matricii, fiecare linie fiind stocată ca un bloc contiguu de memorie, fie prin alipirea coloanelor. Printre limbajele de programare care utilizează stocarea la nivel de linii se numără C, C++, Python, Pascal, iar printre cele în care stocarea se face la nivel de coloană, Fortran, Matlab, R. Pentru a eficientiza aducerea datelor din memorie atunci când se lucrează cu matrici, este indicată parcurgerea acestora pe dimensiunea în care au fost stocate.

2 Sisteme liniare

Fie o matrice $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ și un vector $\boldsymbol{b} \in \mathbb{R}^m$ ce formează un sistem de ecuații liniare, a cărui soluție este $\boldsymbol{x} \in \mathbb{R}^n$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2}$$

Într-un sistem necunoscutele (sau variabilele, parametrii) desemnează gradele de libertate, iar ecuațiile reprezintă constrângerile. În cazul în care numărul de ecuații (m) este mai mic decât cel al necunoscutelor (n), sistemul este subdeterminat. El are mai multe grade de libertate decât constrângeri, prin urmare poate avea o infinitate de soluții. Graficul din stânga din Figura 1 ilustrează acest caz: toate punctele de pe linia verde (singura ecuație a sistemului) sunt soluții. Când $m = n$ sistemul este determinat și are soluție unică. În graficul din mijloc, aceasta este reprezentată ca intersecția celor două drepte (ecuațiile

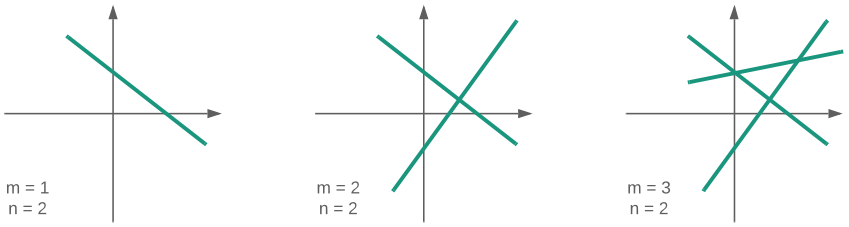

Figura 1: Stânga: sistem subdeterminat, Mijloc: sistem determinat, Dreapta: sistem supradeterminat

sistemului). Iar când numărul constrângerilor îl depășește pe acela al necunoscutelor, sistemul este supradeterminat și nu are soluție. Pentru rezolvarea acestuia, se va căuta o pseudo-soluție sau soluție aproximativă, respectiv un vector $\boldsymbol{x}^*$ care să minimizeze reziduul $\boldsymbol{r} = \boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}$. Când această minimizare implică norma euclidiană $\|\boldsymbol{r}\|_2$ problema se numește problema celor mai mici pătrate (CMMP).

Următorul exemplu [3] ilustrează influența condiționării, definită în Secțiunea 1.1, asupra soluției CMMP. Se consideră un sistem având matricea de tip Hilbert, anume ale cărei elemente au forma

$$H_{i,j} = \frac{1}{i+j-1} \tag{3}$$

Experimentul constă în generarea a 5 sisteme cu dimensiuni diferite, pentru care se cunoaște soluția reală $\boldsymbol{x}_{real}$ și rezolvarea lor în sens CMMP. Pentru moment nu ne vom referi la algoritmii de calcul pentru CMMP, ci doar la problemă în sine. Erorile de aproximare (a soluției) sunt definite ca norma diferenței între soluția reală și cea calculată. Spre comparație, aceeași problemă este aplicată unor sisteme având aceleași dimensiuni, dar pentru care matricea sistemului este o matrice de elemente aleatoare.

Figura 2 prezintă graficele erorilor pentru cele două cazuri. Scala pe axa y este logaritmică. Se observă că în cazul matricilor aleatoare dependența erorii de dimensiunea sistemului este nesemnificativă, iar ușoara creștere a acesteia cu creșterea dimensiunii se datorează acumulării erorilor de rotunjire. Însă în cazul matricilor Hilbert, cunoscute pentru slaba condiționare, eroarea depășește ordinul de mărime al erorilor de rotunjire chiar de la dimensiuni mici ale sistemului ($n = 5$), ajungând la valori extrem de mari pentru $n \geq 11$. (Pentru multe aplicații practice, un sistem de dimensiune $n = 14$ reprezintă un sistem mic.)

O altă concluzie a acestui experiment este recomandarea de a utiliza matrici (și/sau vectori) cu elemente aleatoare atunci când se testează performanțele unui algoritm, pentru a evita astfel de cazuri speciale.

În ce privește metodele de calcul, pentru a găsi soluția sistemului o opțiune o reprezintă rezolvarea ecuației normale, anume

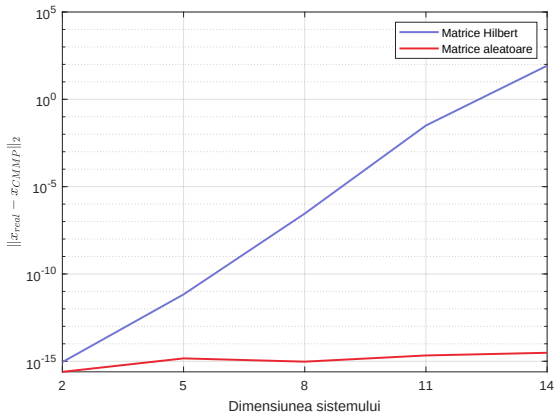

Figura 2: Rezolvarea sistemelor liniare cu o matrice Hilbert și o matrice aleatoare

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \iff \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{b} \Rightarrow \boldsymbol{x} = (\boldsymbol{A}^\top \boldsymbol{A})^{-1} \boldsymbol{A}^\top \boldsymbol{b} \tag{4}$$

Observați că înmulțirea la stânga cu $\boldsymbol{A}^\top$ este necesară în cazul în care matricea nu este pătratică sau este nesingulară, deci nu admite inversare. În schimb, matricea $\boldsymbol{A}^\top \boldsymbol{A}$ este întotdeauna pătratică. Expresia $(\boldsymbol{A}^\top \boldsymbol{A})^{-1}$ se numește pseudo-inversa matricii $\boldsymbol{A}$. Există, însă, metode mai stabile numeric pentru calculul soluției CMMP, instabilitatea fiind dată aici tocmai de calculul pseudo-inversei. O variantă o reprezintă algoritmul QR, ce folosește transformări ortogonale, prezentat în Cursul 2.

3 Regresie liniară

În multe aplicații practice, între mărimile fizice implicate există relații de dependență: turația unui motor depinde de intensitatea curentului aplicat la intrarea acestuia, densitatea unei populații de organisme depinde de variabilele de mediu.

Modelarea presupune găsirea acestor relații dintre variabilele procesului. Regresia liniară reprezintă un tip de model, în care dependența dintre variabile este liniară. Considerăm cazul în care avem doar 2 mărimi, $\boldsymbol{z}$ și $\boldsymbol{w}$ și relația dintre ele $\boldsymbol{w} = f(\boldsymbol{z})$. Variabila $\boldsymbol{z}$ se numește variabila independentă, iar $\boldsymbol{w}$ variabila dependentă. Faptul că relația dintre ele este liniară înseamnă că are forma

$$f(\boldsymbol{z}) = \alpha \boldsymbol{z} + \beta \tag{5}$$

Modelarea presupune, așadar, estimarea valorilor parametrilor α și β. Pentru aceasta este nevoie de un set de date (sau măsurători), adică un set de valori ale variabilelor $\boldsymbol{z}$ și $\boldsymbol{w}$. O dată determinată relația de dependență liniară, pentru orice valoare nouă (care nu există în setul inițial de date) a variabilei $\boldsymbol{z}$, putem

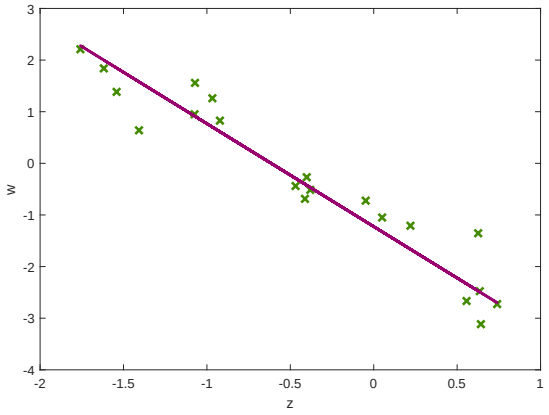

Figura 3: Modelarea relației liniare între două mărimi

calcula valoarea lui $\boldsymbol{w}$, înlocuind în (5) valorile calculate ale parametrilor α și β.

Figura 3 ilustrează un astfel de set de măsurători, în care fiecare punct reprezintă o pereche de valori (z_i, w_i), pentru $i \in [1, m]$, unde m reprezintă dimensiunea setului de date. Relația dintre ele nu este perfect liniară, ceea ce sugerează că datele sunt afectate de zgomot, cum se întâmplă de obicei în practică. Estimarea zgomotului reprezintă o problemă separată, tratăm acum exclusiv modelarea relației dintre variabile.

Dreapta mov este o posibilă astfel de relație liniară. Există și alți candidați viabili, prin urmare modelarea presupune și alegerea unui criteriu de selecție a modelului optim. O opțiune este alegerea dreptei astfel încât suma distanțelor de la puncte la dreaptă să fie minimă. Cu alte cuvinte, se urmărește minimizarea normei erorii

$$\sum_{i=1}^{m}(w_i - f(z_i))^2 = \sum_{i=1}^{m}(w_i - \alpha z_i - \beta)^2 \tag{6}$$

O astfel de abordare echivalează cu problema CMMP, în care se urmărește de asemenea minimizarea erorii. Prin urmare, putem pune problema de regresie în termenii rezolvării unui sistem, în care soluția $\boldsymbol{x}$ o reprezintă ecuația dreptei ce descrie relația liniară dintre variabile. Termenul liber al sistemului $\boldsymbol{Ax} = \boldsymbol{b}$ va fi reprezentat de variabila dependentă, iar matricea sistemului de variabila independentă. Însă pentru a putea modela ecuația de forma (5) este necesară adăugarea unei coloane suplimentare cu elemente egale cu 1 în $\boldsymbol{A}$.

$$\boldsymbol{A} = \begin{bmatrix} z_1 & 1 \\ z_2 & 1 \\ \vdots & \vdots \\ z_m & 1 \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

Sistemul astfel format poate fi rezolvat cu algoritmii cunoscuți pentru CMMP.

4 Ghid Python

Pentru a rezolva exercițiile din laboratorul de astăzi, aveți nevoie de biblioteca `numpy`, de modulul `matplotlib.pyplot` și modulul `scipy.linalg`.

Generarea unei matrici de dimensiune $m \times n$ cu elemente aleatoare se poate realiza utilizând `A = numpy.random.randn(m,n)`.

Pentru a afișa valoarea unei variabile x folosind un număr specificat de zecimale, spre exemplu 2, se folosește instrucțiunea `print("%.2f" % x)`. Alternativ, pentru versiuni de Python mai noi de 3.6, se poate utiliza `f'{x:.2f}'`.

Biblioteca `scipy` conține o funcție pentru rezolvarea problemei CMMP, `lstsq()`,

`x, _, _, _ = linalg.lstsq(A, b).`

Notația "`_`" semnalează faptul că respectiva mărime returnată de funcție nu se memorează într-o variabilă.

Pentru a încărca un conținutul unui fișier `.csv` utilizați

`res = np.genfromtxt ('file.csv', delimiter=",").`

În Python indexarea elementelor unui vector sau a unei matrici începe de la 0. Pentru a selecta toate elementele celei de-a doua coloane dintr-o matrice, folosiți `col2 = A[:,1]`.

Dacă ați importat modulul `matplotlib.pyplot` cu numele `plt`, puteți afișa grafic un punct specificându-i coordonatele pe axele x și y, folosind sintaxa: `plt.plot(x, y, marker="x", markersize=8, markerfacecolor="green")`.

5 Exerciții

1. Fie un șir de numere definit astfel [3]

$$x_{k+1} = \begin{cases} 2x_k & , x_k \in [0, 0.5) \\ 2x_k - 1 & , x_k \in (0.5, 1] \end{cases} \tag{7}$$

   (a) Considerând valoarea inițială $x_0 = 0.1$, ce valoare va avea x la pasul $k = 60$?

   (b) Scrieți o procedură care să calculeze șirul pentru un număr dat de iterații. Verificați dacă obțineți același rezultat ca la punctul anterior.

(c) Afișați valorile lui x la fiecare iterație cu o precizie de 4 zecimale și comentați rezultatele. De unde apar diferențele față de valorile calculate pe hârtie?

2. Scrieți o procedură care utilizează eliminarea Gaussiană pentru a rezolva un sistem determinat $\boldsymbol{Ax} = \boldsymbol{b}$.

   (a) Aducerea sistemului la forma superior triunghiulară. Folosiți codul din Laboratorul Precedent, unde matricea $\boldsymbol{A}$ este adusă la formă superior triunghiulară utilizând eliminarea Gaussiană. Pentru ca sistemul rezultat să fie echivalent cu cel inițial este nevoie, în plus, de transformarea termenului liber $\boldsymbol{b}$ în mod similar cu cea a lui $\boldsymbol{A}$. Introducerea zerourilor în pozițiile subdiagonale ale matricii pe coloana k presupune modificarea liniilor $k+1 : n$ ale sistemului. Pentru o astfel de linie i, elementul corespunzător din termenul liber are forma

   $$\boldsymbol{b}_i = \boldsymbol{b}_i + \boldsymbol{b}_k \cdot \boldsymbol{a}_{i,k} \tag{8}$$

   unde în $\boldsymbol{a}_{i,k}$ a fost suprascris multiplicatorul Gaussian ce introduce 0-ul în poziția (i, k) a matricii.

   (b) Implementați o procedură pentru rezolvarea sistemelor superior triunghiulare prin metoda substituției (algoritmul UTRIS), pornind de la codul algoritmului LTRIS din Cursul 1.

   (c) Testați soluția pe un sistem de dimensiune $n = 6$, în care matricea $\boldsymbol{A}$ are elemente aleatoare. Utilizați procedura de la punctul b) pentru a rezolva sistemul obținut la punctul a). Verificați rezultatul numeric obținut.

3. Fișierul `regresie.csv` conține setul de măsurători prezentate în Figura 3. Prima coloană reprezintă variabila $\boldsymbol{z}$, iar cea de-a doua $\boldsymbol{w}$. Rezolvați problema de regresie liniară (calculați parametrii dreptei):

   (a) Construiți matricea sistemului $\boldsymbol{A}$ din variabila independentă $\boldsymbol{z}$ și termenul liber din variabila dependentă $\boldsymbol{w}$.

   (b) Calculați soluția CMMP (ecuația dreptei) utilizând funcția `lstsq()`.

   (c) Afișați grafic punctele corespunzătoare măsurătorilor, cât și dreapta de regresie obținută.

Bibliografie

[1] BLAS Package. http://www.netlib.org/blas/.

[2] LAPACK Package. http://www.netlib.org/lapack/.

[3] Anne Greenbaum and Tim P. Chartier. *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*. Princeton University Press, 2012.