ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΜΟΣ Ι

ΔΙΑΛΕΞΗ 11 (μέρος 2) 20/11/09

Ε. ΓΑΛΛΟΠΟΥΛΟΣ
Τμήμα Μηχ. Η/Υ και Πληροφορικής
Παν/μιο Πατρών

ΑΓΑ.1 *Μας δίνεται* $A \in \mathbb{R}^{n \times n}$, *και* $b \in \mathbb{R}^n$. *Θέλουμε να υπολογίσουμε* $x \in \mathbb{R}^n$ *τέτοιο ώστε* $Ax = b$.

Παρατηρήσεις: Το γενικότερο πρόβλημα είναι η εύρεση του $\arg\min_{x \in \mathbb{R}^n} \|b - Ax\|$ που επιτρέπει τη διαχείριση μη τετραγωνικών ή και μη ομαλών προβλημάτων. Η ευκλείδεια νόρμα αντιστοιχεί στο πρόβλημα ελαχίστων τετραγώνων (κεφ. 6).

Θεώρημα 1 (Κανόνας του Cramer**).** *Έστω $A \in \mathbb{R}^{n\times n}$ και $b \in \mathbb{R}^n$. Τα στοιχεία ξ_j $(1 \leq j \leq n)$ του διανύσματος $x \in \mathbb{R}^n$ που λύνουν το σύστημα $Ax = b$ ικανοποιούν τις σχέσεις:*

$$\xi_i = \frac{\det(A(i|b))}{\det(A)}, i = 1, \ldots, n.$$

όπου $A(i|b)$ είναι το μητρώο που προκύπτει αν αντικαταστήσουμε την i στήλη του A με το b. □

Μη πρακτικός Υψηλή πολυπλοκότητα και μη εξασφαλισμένο σφάλμα.
Υπενθύμιση: Στη συντριπτική πλειοψηφία των περιπτώσεων ΔΕΝ θέλουμε να υπολογίσουμε το αντίστροφο (ακόμα και αν το γράφουμε ή το «αναφέρουμε»).
Γιατί (1); Αυτό που ενδιαφέρει είναι να υπολογίσουμε το $Q = A^{-1}B$ ή $U = CA^{-1}$ ή γενικότερα $CA^{-1}B$ για δεδομένα B, C, όπου B έχει n γραμμές και το C έχει n στήλες.
Γιατί (2); Όπως αν θέλουμε να υπολογίσουμε το β/α για βαθμωτούς, κάνουμε διαίρεση και ΔΕΝ υπολογίζουμε πρώτα $\gamma := 1/\alpha$ και μετά $\gamma\beta$.
Γιατί (3); Το A^{-1} συνήθως δεν έχει κάποια ειδική δομή. Π.χ. αν το A τριδιαγώνιο (άρα αποθήκευση σε $\approx 3n$ θέσεις), το A^{-1} πυκνό (άρα αποθήκευση σε $\approx n^2$ θέσεις).
Γιατί (4); Ο υπολογισμός του A^{-1} στοιχίζει λίγο περισσότερο από το να λύσουμε ένα

γραμμικό σύστημα.

Για $X = A^{-1}B$, αν $B \in \mathbb{R}^{n\times k}$ αρκεί να λύσουμε $AX = B$, δηλ. να λύσουμε k γραμμικά συστήματα με το ίδιο μητρώο συντελεστών οπότε θα αρκέσει μια μόνο παραγοντοποίηση, π.χ. LU.

Για $Y = CA^{-1}$, αν $C \in \mathbb{R}^{m\times n}$ έχουμε $YA = C \Rightarrow A^\top Y^\top = C^\top$, επομένως λύνουμε ως προς $Y^\top$ τα m γραμμικά συστήματα με το ίδιο μητρώο συντελεστών, $A^\top$, οπότε πάλι αρκεί μια μόνο παραγοντοποίηση.

Για $Z = CA^{-1}B$, θα μπορούσαμε να υπολογίσουμε $Z = C(A^{-1}B) = CX$ ή $Z = (A^{-\top}C^\top)B = Y^\top B$. Η σειρά υπολογισμών που ελαχιστοποιεί τις πράξεις εξαρτάται από τις σχέσεις μεταξύ των m, k, n.

$$
\begin{aligned}
A &= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \\
&= \begin{pmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & \underbrace{A_{22} - A_{21}A_{11}^{-1}A_{12}}_{\text{συμπλήρωμα Schur}} \end{pmatrix} \\
A^{-1} &= \begin{pmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12}S^{-1} \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -A_{21}A_{11}^{-1} & I \end{pmatrix} \\
&= \begin{pmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}S^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}S^{-1} \\ -S^{-1}A_{21}A_{11}^{-1} & S^{-1} \end{pmatrix}
\end{aligned}
$$

Για το αντίστροφο χρειαζόμαστε:

1. Το A_{11}^{-1},
2. τα $A_{11}^{-1}A_{12}$ και $A_{21}A_{11}^{-1}$,
3. το $S^{-1} = (A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}$,
4. τα $-S^{-1}(A_{21}A_{11}^{-1})$, $(A_{11}^{-1}A_{12})S^{-1}$,
5. και το $A_{11}^{-1} + (A_{11}^{-1}A_{12})S^{-1}(A_{21}A_{11}^{-1})$

Συνολικά

- 2 αντιστροφές μητρώων μεγέθους $n/2$, και
- 6 πολλαπλασιασμούς μητρώων μεγέθους $n/2$,
- 2 προσθέσεις μητρώων μεγέθους $n/2$, δηλ.

Επομένως $\boxed{T_{\text{INV}}(n) = 2T_{\text{INV}}(n/2) + 6T_{\text{MUL}}(n/2) + 2T_{\text{ADD}}(n/2)}$ και έπεται ότι

Θεώρημα 2. *Αν το μητρώο A είναι αντιστρέψιμο, μπορούμε να υπολογίσουμε το αντίστροφό του με λιγότερες από $5.64n^{\log 7}$ αριθμητικές πράξεις.* □

ΣΥΜΠΕΡΑΣΜΑ: Αν διαθέτουμε έναν υπερταχύ ($\Omega = O(n^{2+\mu})$ για $\mu < 1$), αλγόριθμο πολλαπλασιασμού μητρώων, μπορούμε να τον χρησιμοποιήσουμε άμεσα και να κατασκευάσουμε αλγόριθμο αντιστροφής με ίδια ασυμπτωτική πολυπλοκότητα!

ΠΡΟΣΟΧΗ

Για να είναι εφικτός ο παραπάνω υπερταχύς αλγόριθμος θα πρέπει να είναι αντιστρέψιμα τα υπομητρώα που χρησιμοποιούμε μ' αυτόν τον τρόπ κατά τη διάρκεια της αναδρομικής επίλυσης:
π.χ. στο πρώτο βήμα παραπάνω, τα A_{11} και $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$.

Η αντιστρεψιμότητα του A δεν εξασφαλίζει την αντιστρεψιμότητα των παραπάνω!

Για παράδειγμα, το μητρώο $[0, 1; 1, 1]$ είναι αντιστρέψιμο αλλά δεν ικανοποιεί την παραπάνω συνθήκη καθώς $\alpha_{1,1} = 0$.
Ένας τρόπος να αποφύγουμε το παραπάνω πρόβλημα είναι αντί για το $Ax = b$ να επιλύσουμε το ισοδύναμο σύστημα $A^\top Ax = A^\top b$. Αν το A είναι αντιστρέψιμο, τότε το $A^\top A$ είναι και αυτό αντιστρέψιμο και μπορούμε να αποδείξουμε ότι όλα τα υπομητρώα που εμφανίζονται στην αναδρομή και χρειάζονται αντιστροφή είναι αντιστρέψιμα.
Πώς το δείχνουμε (ΠΡΟΑΙΡΕΤΙΚΟ); Δύο σκέλη στην απόδειξη· δείχνουμε ότι:

1. το μητρώο $A^\top A$ είναι συμμετρικό θετικά ορισμένο (ΣΘΟ).
2. τα μητρώα α) $A_{1,1}$ και β) S που προέρχονται από ένα ΣΘΟ μητρώο A σύμφωνα με την παραγοντοποίηση κατά ορμαθούς

$$A = \begin{pmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & S \end{pmatrix}$$

τεμαχισμό είναι επίσης ΣΘΟ.

- Σχετικά με το (1): Αν $x \neq 0$ τότε $x^\top A^\top A x = (Ax)^\top (Ax) > 0$ από τον ορισμό του εσωτερικού γινομένου.
- Σχετικά με το (2α): Έστω ότι $x = [x_1; 0]^\top$ όπου το μέγεθος του x_1 είναι ίδιο με τον αριθμό γραμμών/στηλών του $A_{1,1}$. Τότε $0 < x^\top A x = x_1^\top A_{1,1} x_1$.
- Σχετικά με το (2β): Έστω $x = [x_1; x_2]^\top$, τότε

$$\begin{aligned} 0 \;&<\; x^\top A x = x_1^\top A_{1,1} x_1 + x_1^\top A_{1,2} x_2 + x_2^\top A_{2,1} x_1 + x_2^\top A_{2,2} x_2 \\ &<\; (x_1 + A_{1,1}^{-1} A_{1,2} x_2)^\top A_{1,1} (x_1 + A_{1,1}^{-1} A_{1,2} x_2) + x_2^\top (A_{2,2} - A_{2,1} A_{1,1}^{-1} A_{1,2}) x_2 \end{aligned}$$

για οποιαδήποτε επιλογή των x_1, x_2 αρκεί $x \neq 0$. Επομένως, αν διαλέξουμε x_1, x_2 έτσι ώστε $x_1 + A_{1,1}^{-1} A_{1,2} x_2 \equiv 0$ τότε

$$x_2^\top (A_{2,2} - A_{2,1} A_{1,1}^{-1} A_{1,2}) x_2 = x_2^\top S x_2 > 0$$

και προκύπτει ότι το S ενός ΣΘΟ μητρώου επίσης είναι ΣΘΟ.

Είδη μητρώων

Αλγεβρικές ιδιότητες:

- A γενικό
- $A = A^*$, όπου A^* συμβολίζει το «συζυγές ανάστροφο» του A (αν $A \in \mathbb{R}^{n\times n}$ προφανώς $A^\top = A^*$.
- $AA^* = A^*A$
- $\forall x \neq 0, x^*Ax > 0$.
- $A^*A = I$

Δομή

- τριγωνική, τριδιαγώνια, Hessenberg
- συμμετρικό$A = A^T$, ερμιτιανό $A = A^*$
- A Toeplitz, Hankel, Vandermonde, circulant

Ως προς αριθμ. μηδενικών και αποθήκευση

- πυκνό
- αραιό

ποικιλία αραιών τρόπων αποθήκευσης

Παρατηρήσεις

Συμπίεση: Πόσες πράξεις χρειάζονται για να εκτελεσθεί ο πολλαπλασιασμός Ab;

αποτελεσματικότητα $\rightarrow$ *χρήση πληροφορίας*

Παράδειγμα 1. *Επιτρέπουν ταχύτερες μεθόδους διαχείρισης:* FFT, circulant, Toeplitz
□

Θεμελιώδης τεχνική επίλυσης: Μέθοδοι παραγοντοποίησης

Εξαιρετικά σημαντική κατηγορία τεχνικών επίλυσης των βασικών προβλημάτων της υπολογιστικής γραμμικής άλγεβρας (κεφάλαια 5, 6, 7 και αλλού) γραμμικών συστημάτων αλλά και άλλων προβλημάτων αυτής της κατηγορίας λέγεται

matrix decomposition

στην οποία θα αναφερόμαστε ως

διάσπαση ή παραγοντοποίηση μητρώου

- Οι περισσότερες τεχνικές παραγοντοποίησης αναπτύχθηκαν ανεξάρτητα η μια από την άλλη
- η σύνδεσή τους και ενοποιημένη παρουσίασή τους οφείλεται στον Alston Householder (1904-1993)
- ολοκληρωμένη «εγκυκλοπαιδική» προσέγγιση στο βιβλίο του G.W. Stewart *Matrix Decompositions* (2002).

Η κεντρική ιδέα είναι, δοθέντος του $A \in \mathbb{R}^{m\times n}$, να προσπαθήσουμε να υπολογίσουμε κατάλληλους παράγοντες C, D, E, τέτοιους ώστε

$$A = CDE \text{ ή γενικότερα } \quad A \approx CDE$$

όπου οι παράγοντες C, D, E είναι «απλούστεροι» αλλά εμπεριέχουν ό,τι χρειάζεται για το A.

Ποια παραγοντοποίηση επιλέγουμε;

Γενικά κριτήρια:

- το πρόβλημα υπό λύση (π.χ. ΑΓΑ.2, ΑΓΑ.3, κ.λπ.)
- τα αλγεβρικά χαρακτηριστικά του μητρώου (συμμετρικό ορισμένο, συμμετρικό ημιορισμένο, ...)
- τα δομικά χαρακτηριστικά του μητρώου (μέγεθος, πυκνότητα, ειδική μορφή (π.χ. Toeplitz), ...)
- το Υ/Σ (σειραϊκό, παράλληλο, ...)
- σε μεγάλα προβλήματα αναζητούμε οικονομικές προσεγγιστικές παραγοντοποιήσεις

Πλεονεκτήματα της μεθοδολογίας παραγοντοποίησης[1]

1. Ένα είδος παραγοντοποίησης μπορεί να εφαρμοστεί σε πολλών ειδών προβλήματα.
2. Μια παραγοντοποίηση μπορεί να επαναχρησιμοποιηθεί όταν το μητρώο παραμένει το ίδιο.
3. Η μεθοδολογία παραγοντοποίησης ενοποιεί πολλούς αλγορίθμους που συχνά υπολογίζουν το ίδιο αντικείμενο
4. Διευκολύνεται η ανάλυση σφάλματος α.κ.υ.
5. Πολλές παραγοντοποιήσεις προσφέρουν τη δυνατότητα οικονομικής ανανέωσης.
6. Επικεντρωνόμενοι σε λίγες παραγοντοποιήσεις που αποτελούν κοινό παρονομαστή πολλών ειδικότερων προβλημάτων, διευκολύνεται η οργάνωση και σχεδίαση σημαντικών βιβλιοθηκών.

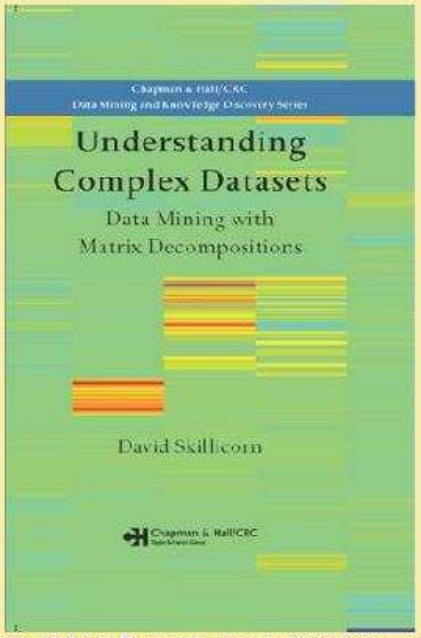
Αποφεύγουμε να υπολογίσουμε το γινόμενο, προτιμούμε τους παράγοντες![2]

[1] Από Stewart'00

«Αναρίθμητες» και νέες εφαρμογές

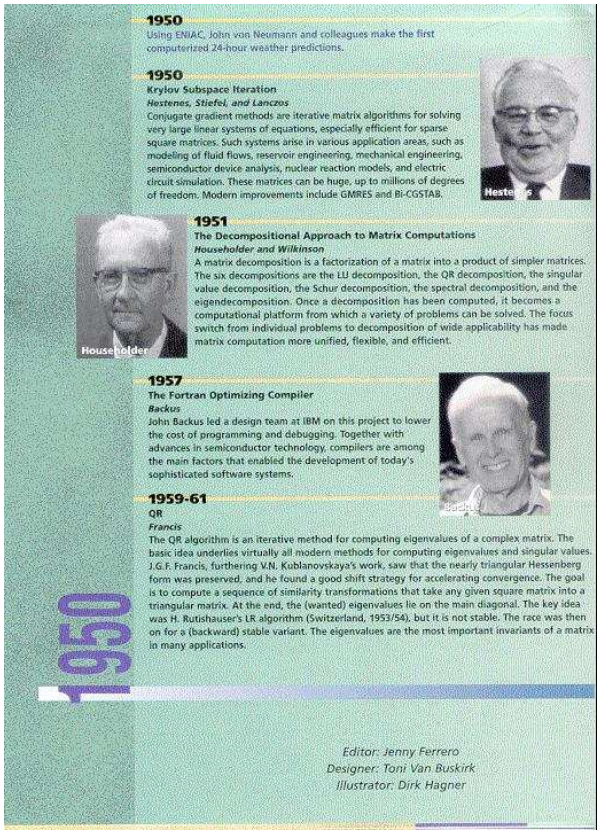
---

on of Anomaly Detection in Graphs (pdf).



my new book: Understanding Complex Datasets: Data Mining using Matrix Decompositions, CRC Press, available June 2007.

Μερικές σημαντικές διασπάσεις/παραγοντοποιήσεις μητρώων



LU, PLU, LDU	
$LL^\top, LDL^\top$	Cholesky
QR, QRP	παραγοντοποίηση QR
$V\Lambda V^\top$, $V\Lambda V^{-1}$	φασματική
$VTV^\top$	παραγοντοποίηση Schur
$U\Sigma V^\top$	SVD
QM	πολική
CD, $C, D \geq 0$	μη αρνητική (NMF)

- Δείτε τον «κατάλογο παραγοντοποιήσεων» στο βιβλίο του Strang (Εκδ. ΠΠ), ς. 693-695.
- Ενδιαφέροντα (και κάποια ιστορικά) στοιχεία στο άρθρο[3] του Stewart (αναφέρεται στις «The BIG SIX» παραγοντοποιήσεις) The Decompositional Approach to Matrix Computation, IEEE CS&E Magazine, 2000.

[3] http://portal.acm.org/citation.cfm?id=615766 και σύνδεσμο από το ΗΜΕΡΟΛΟΓΙΟ

$A = LU$ όπου L, U κάτω και άνω τριγωνικά αντίστοιχα.

$A = QR$ όπου Q, R ορθογώνιο και άνω τριγωνικό αντίστοιχα.

$A = Q\Lambda Q^{-1}$ όπου Λ διαγώνιο.

Παραδείγματα Έστω $Ax = b$ τότε $Q^{-1}AQQ^{-1}x = Q^{-1}b$ επομένως $\Lambda y = \hat{b}$. Επομένως ένας τρόπος επίλυσης είναι να κάνουμε τα εξής βήματα:

1. Υπολογισμός των Q, Λ. Σημ. αυτό γίνεται μόνο μια φορά ανεξάρτητα από τον αριθμό των συστημάτων που ενδεχομένως πρέπει να λυθούν. Κόστος C_0.
2. Υπολογισμός του $\hat{b} := Q^{-1}b$, κόστος C_1.
3. Υπολογισμός του $\hat{x} := \Lambda^{-1}\hat{b}$, κόστος n αρ. πράξεις.
4. Υπολογισμός του $x = Q\hat{x}$, κόστος C_2.

Εφόσον το κόστος της επίλυσης με κλασικό τρόπο (π.χ. Gauss) είναι $O(n^3)$, θέλουμε

$$C_0 + C_1 + C_2 + O(n) < O(n^3)$$

Έστω ότι τα Q, Λ είναι ήδη γνωστά (άρα $C_0 = 0$. Τότε το ανωτέρω ισχύει αν $C_1, C_2 = O(n^2)$)

Ταξινόμηση κόστους για την επίλυση κοινών συστημάτων

Χωρίς να πούμε τίποτα για τους αλγόριθμους επίλυσης αλλά υποθέτοντας ότι δεν χρησιμοποιούμε υπερταχείες μεθόδους τύπου Strassen, ισχύει η παρακάτω ταξινόμηση:

Είδος μητρώου	Ω	**Είδος μητρώου**	Ω
διαγώνιο	$O(n)$	διδιαγώνιο	$O(n)$
μητρώο ζώνης εύρους $\beta = p + q$	$O(npq)$	τριγωνικό	$O(n^2)$
ορθογώνιο	$O(n^2)$	Hessenberg	$O(n^2)$
ΣΘΟ	$\frac{1}{3}n^3 + O(n^2)$	γενικό	$\frac{2}{3}n^3 + O(n^2)$
μη αρνητικά	NP-πλήρης		

- Ο παραπάνω πίνακας δεν είναι πλήρης
- δυο σημαντικές κατηγορίες που είναι απούσες είναι
  - { γενικά αραιά μητρώα (sparse matrices),
  - { πυκνά δομημένα μητρώα (dense structured matrices).
- Για τις κατηγορίες αυτές υπάρχουν ειδικές μέθοδοι που αξιοποιούν τα ειδικά χαρακτηριστικά.
- Προσοχή: Οι δυο αυτές τελευταίες κατηγορίες εμφανίζονται και αυτές συχνά στις εφαρμογές (θυμηθείτε τα λόγια του Turing στο ACE Report.

LU στη MATLAB 5.3

```
[L,U] = LU(X) stores a upper triangular
matrix in U and a "psychologically lower
triangular matrix", i.e.  a product
of lower triangular and permutation
matrices, in L , so that X = L*U.
[L,U,P] = LU(X) returns lower triangular
matrix L, upper triangular matrix U, and
permutation matrix P so that P*X = L*U.
By itself, LU(X) returns the output from
LINPACK'S ZGEFA routine.
```

LU στη MATLAB 6

```
[L,U] = LU(X) stores an upper triangular
matrix in U and a "psychologically lower
triangular matrix" (i.e.  a product
of lower triangular and permutation
matrices) in L, so that X = L*U. X can
be rectangular.
[L,U,P] = LU(X) returns lower triangular
matrix L, upper triangular matrix U, and
permutation matrix P so that P*X = L*U.
LU(X), with one output argument, returns
the output from LAPACK'S DGETRF or ZGETRF
routine.
```

LINPACK benchmark (Jack Dongarra)

Επίλυση συστημάτων $n = 100, 1000$. Δείτε στο

`http://www.netlib.org/benchmark/top500.html`

> The benchmark used in the LINPACK Benchmark is to solve a dense system of linear equations. For the TOP500, we used that version of the benchmark that allows the user to scale the size of the problem and to optimize the software in order to achieve the best performance for a given machine. This performance does not reflect the overall performance of a given system, as no single number ever can. It does, however, reflect the performance of a dedicated system for solving a dense system of linear equations. Since the problem is very regular, the performance achieved is quite high, and the performance numbers give a good correction of peak performance.