Înmulțirea matricelor

Bejan Ștefan

Mai 2024

1 Definiții și notații

1.1 Definiția matricelor și operațiile de bază cu matrice

Definiție.
Se numește matrice cu m linii si n coloane (de tip $m \times n, m, n \in \mathbb{N}$) un tablou cu m linii si n coloane:

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{m1} & a_{m2} & ... & a_{mn} \end{pmatrix},$$

ale cărui elemente a_{ij} sunt numere complexe și $i = \overline{1, m}, j = \overline{1, n}$.
Egalitatea a două matrice.
Fie $A = (a_{ij})$, $B = (b_{ij}) \in M_{m,n}(C)$.Se spune că matricele A, B sunt egale și se scrie $A = B$ dacă $a_{ij} = b_{ij}, \forall i, j \in \mathbb{N}$.
Adunarea matricelor.
Fie $A = (a_{ij}), B = (b_{ij}), C = (c_{ij}) \in M_{m,n}(\mathbb{C})$.
Matricea C se numește suma matricelor A, B dacă:
$c_{ij} = a_{ij} + b_{ij}, \forall i, j \in \mathbb{N}$.
Observații:
1. Două matrice se pot aduna dacă sunt de același tip, adică au același număr de linii și același număr de coloane, deci $A, B \in M_{m,n}(\mathbb{C})$.
2. Explicit, adunarea matricelor A, B înseamnă:

$$\begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{m1} & a_{m2} & ... & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & ... & b_{1n} \\ b_{21} & b_{22} & ... & b_{2n} \\ ... & ... & ... & ... \\ b_{m1} & b_{m2} & ... & b_{mn} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & ... & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & ... & a_{2n}+b_{2n} \\ ... & ... & ... & ... \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & ... & a_{mn}+b_{mn} \end{pmatrix}.$$

Proprietăți ale adunării matricelor:
1. Asociativitatea. Adunarea matricelor este asociativă, adică:
$(A + B) + C = A + (B + C), \forall A, B, C \in M_{m,n}(\mathbb{C})$.
2. Comutativitatea. Adunarea matricelor este comutativă, adică:
$A + B = B + A, \forall A, B \in M_{m,n}(\mathbb{C})$.
3. Element neutru. Adunarea matricelor admite matricea nulă ca element neutru, adică:
$\exists O_{m,n} \in M_{m,n}(\mathbb{C})$ astfel încât $A + O_{m,n} = A, \forall A \in M_{m,n}(\mathbb{C})$.
4. Element opus. Orice matrice $A \in M_{m,n}(\mathbb{C})$ are un opus, notat $-A$, astfel încât:
$A + (-A) = O_{m,n}$.
Înmulțirea cu scalari a matricelor.
Fie $\lambda \in \mathbb{C}$ și $A = (a_{ij}) \in M_{m,n}(\mathbb{C})$. Se numește produsul dintre scalarul λ și matricea A, matricea notată $\lambda A \in M_{m,n}(\mathbb{C})$ definită prin $\lambda A = (\lambda a_{ij})$.
Observații:
A înmulți o matrice cu un scalar revine la a înmulți toate elementele matricei cu acest scalar. Deci:

$$\lambda A = \begin{pmatrix} \lambda a_{11} & \lambda a_{12} & ... & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & ... & \lambda a_{2n} \\ ... & ... & ... & ... \\ \lambda a_{m1} & \lambda a_{m2} & ... & \lambda a_{mn} \end{pmatrix}.$$

Proprietăți ale înmulțirii matricelor cu scalari:
$\lambda(\rho A) = (\lambda\rho)A, \forall\lambda, \rho \in \mathbb{C}, A \in M_{m,n}(\mathbb{C})$.
$\lambda(A + B) = \lambda A + \lambda B, \forall\lambda \in \mathbb{C}$ și $A, B \in M_{m,n}(\mathbb{C})$.
$(\lambda + \rho)A = \lambda A + \rho A$, $\forall\lambda, \rho \in \mathbb{C}$, $A \in M_{m,n}(\mathbb{C})$.
$1 \times A = A$, $1 \in \mathbb{C}$, $A \in M_{m,n}(\mathbb{C})$.

2 Aplicațiile matricelor în viața cotidiană

2.1 Contextul și importanța înmulțirii matricelor în diverse domenii

Înmulțirea matricelor este o operație fundamentală în algebra liniară și are numeroase aplicații importante în matematică, inginerie și informatică. Iată câteva exemple despre cum este utilizată înmulțirea matricelor în diferite domenii:

1) **Transformările liniare**: Înmulțirea matricelor este folosită pentru a reprezenta și aplica transformările liniare în grafică computerizată, robotică, fizică și alte domenii. O matrice poate reprezenta o transformare a unui vector, cum ar fi o rotație, scalare sau translatație.

Exemplu: Considerăm un vector bidimensional $v = \begin{pmatrix} x \\ y \end{pmatrix}$. Pentru a roti acest vector cu un unghi θ în sens trigonometric (în sensul acelor de ceasornic), putem folosi următoarea matrice de transformare:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix},$$

unde x' și y' sunt componentele vectorului rotit. Această matrice de transformare realizează o rotație a vectorului v în jurul originii cu un unghi θ.

2) **Rezolvarea sistemelor de ecuații liniare**: Înmulțirea matricelor poate fi folosită pentru a rezolva sisteme de ecuații liniare, care sunt întâlnite frecvent în inginerie, fizică, economie și alte domenii. Prin înmulțirea coeficienților variabilelor cu o matrice, putem determina valorile acestor variabile.

3) **Lanțurile Markov**: Înmulțirea matricelor este folosită pentru a analiza lanțurile Markov, care sunt modele matematice ale sistemelor care trec între diferite stări în timp. Prin înmulțirea unei matrice de tranziție cu un vector de probabilitate, putem calcula probabilitatea de a fi în fiecare stare la un moment dat.

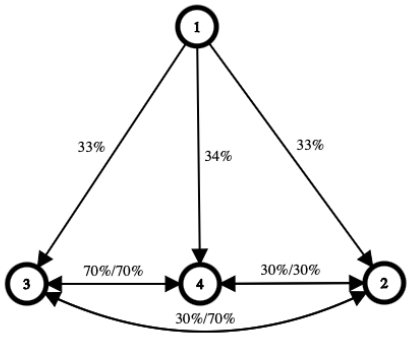
Exemplu: Să presupunem că avem următoarele cuvinte posibile care pot urma după "Cine": "urmează", "vorbește" și "ascultă". Prin natura sa, la început, sistemul se va afla în mod trivial în starea asociată cuvântului "Cine". Așadar, vectorul de probabilitate inițial va arăta așa:

$$p_{inițial} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}.$$

Pasul următor este de a ne construi matricea de tranziție. Acesta presupune stabilirea probabilităților de a trece din starea i în starea j:

$$M_{tranziție} = \begin{pmatrix} 0 & 0.33 & 0.33 & 0.34 \\ 0 & 0 & 0.7 & 0.3 \\ 0 & 0.3 & 0 & 0.7 \\ 0 & 0.3 & 0.7 & 0 \end{pmatrix}.$$

Bonus: Această matrice poate fi reprezentată precum un graf orientat. Se vor codifica cuvintele astfel: "Cine" $\rightarrow$ 1, "urmează" $\rightarrow$ 2, "vorbește" $\rightarrow$ 3 și "ascultă" $\rightarrow$ 4:



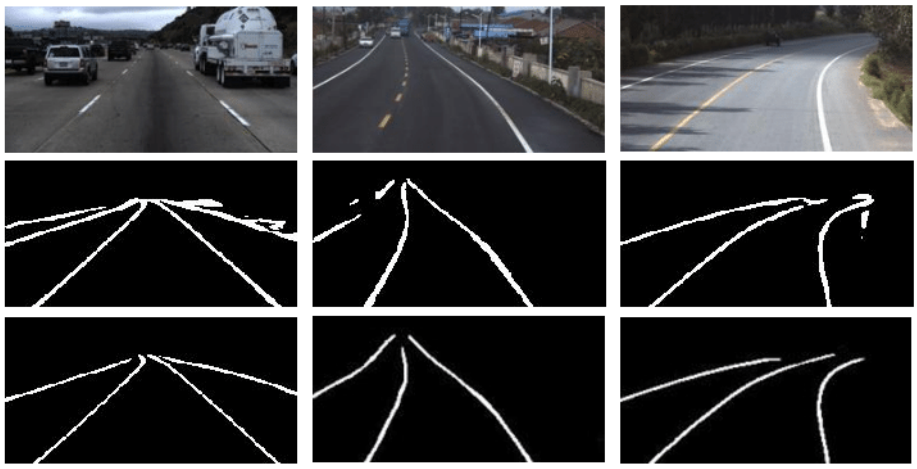
Înmulțind vectorul de stare și matricea de transformare, vom obține probabilitatea de apariție pentru fiecare cuvânt după o tranziție:

$$p_{următor} = p_{inițial} \cdot M_{tranziție} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0.33 & 0.33 & 0.34 \\ 0 & 0 & 0.7 & 0.3 \\ 0 & 0.3 & 0 & 0.7 \\ 0 & 0.3 & 0.7 & 0 \end{pmatrix} \Rightarrow$$

$$p_{următor} = \begin{pmatrix} 0 & 0.33 & 0.33 & 0.34 \end{pmatrix}$$

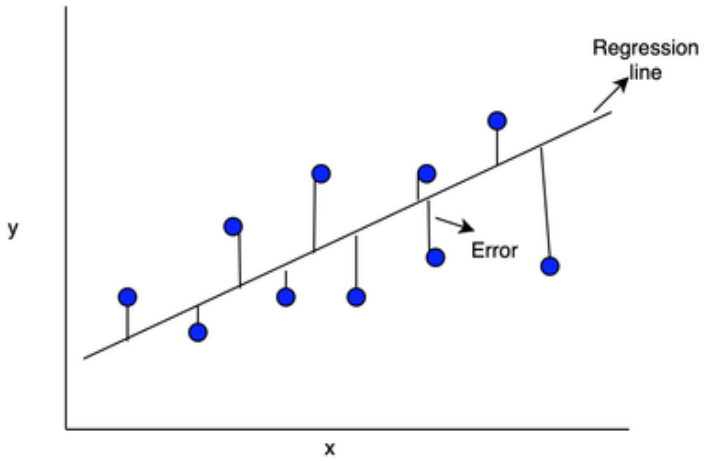
Conform $p_{următor}$, cuvântul "urmează" are o probabilitate de 33 %, "vorbește" tot de 33%, iar "ascultă" de 34%. Într-un mod recursiv, se pot calcula și următorii vectori de stări pentru viitoarele tranziții.

4) **Prelucrarea imaginilor**: Înmulțirea matriceală este folosită în prelucrarea imaginilor pentru a aplica filtre și transformări imaginilor. O matrice poate reprezenta un nucleu de convoluție, care poate fi înmulțit cu o matrice de imagine pentru a aplica un filtru.
Exemplu: Detectorul de benzi albe pentru mașini este un exemplu comun de utilizare a prelucrării imaginilor în viața de zi cu zi. În acest caz, imaginea captată de o cameră de pe mașină poate fi reprezentată sub formă de matrice, unde fiecare element al matricei corespunde unui pixel din imagine. Pentru a detecta benzile albe de pe șosea, se aplică un filtru de prelucrare a imaginii, cum ar fi un filtru de detectare a marginilor sau un filtru de convoluție specific. Acest filtru este reprezentat de o matrice, care este apoi multiplicată cu matricea imaginii pentru a evidenția caracteristicile relevante, cum ar fi marginile benzilor albe. Înmulțind matricea filtrului cu matricea imaginii pixel cu pixel, se obține o nouă matrice care reprezintă imaginea procesată, în care benzile albe sunt evidențiate. Acest lucru permite apoi algoritmului de detectare să identifice poziția și direcția benzilor albe, ajutând astfel mașina să rămână pe traiectoria corectă în timpul conducerei.



5) **Învățarea automată**: Înmulțirea matricelor este folosită extensiv în algoritmii de învățare automată, precum regresia liniară, rețelele neurale și analiza componentelor principale. Matricele sunt folosite pentru a reprezenta seturile de date și parametrii modelelor, iar înmulțirea matricelor este folosită pentru a actualiza și aplica modelele.
Exemplu: Pentru a efectua analiza regresiei liniare într-un mod eficient și sistematic, datele sunt organizate sub formă de matrice, ceea ce permite utilizarea unor operații matematice convenabile pentru estimarea coeficienților modelului. În acest context, variabilele explicative sunt dispuse într-o matrice de design, iar variabila de răspuns este reprezentată printr-un vector. Astfel, prin aplicarea operațiilor de înmulțire a matricelor, regresia liniară determină relația liniară între variabilele explicative și variabila de răspuns, păstrând în același timp

structura și integritatea datelor inițiale.



2.2 Motivația pentru studierea complexităților operațiilor matricelor

Studiul complexității operațiilor matricelor este crucial din două motive principale: eficiența algoritmilor și optimizarea performanței. În primul rând, înțelegerea complexității operațiilor matricelor ne permite să evaluăm și să comparăm algoritmii pentru a determina cel mai eficient într-un anumit context. Algoritmii cu o complexitate mai mică sunt de obicei mai rapizi și consumă mai puține resurse, ceea ce poate fi crucial în aplicații practice. În al doilea rând, cunoașterea complexității operațiilor matricelor ne ajută să optimizăm performanța algoritmilor noștri, astfel încât să putem face față eficient unor volume mari de date sau să reducem timpul necesar pentru a efectua calculele.

3 Algoritm clasic de înmulțire a matricelor

3.1 Metodă clasică de înmulțire a matricelor

În matematică, în special în algebra liniară, înmulțirea matricilor sau înmulțirea matriceală este o operație binară care produce o matrice din două matrici. La înmulțirea matriceală, numărul de coloane din prima matrice trebuie să fie egal cu numărul de linii din a doua matrice. Matricea rezultată, cunoscută sub numele de produs matriceal, are numărul de linii ale primei matrice și numărul de coloane ale celei de-a doua matrice. Înmulțirea matricială a fost descrisă pentru prima dată de matematicianul francez Jacques Philippe Marie Binet în 1812, pentru a reprezenta compunerea funcțiilor de transformări liniare care sunt reprezentate cu ajutorul matricilor. Astfel, înmulțirea matricială a devenit

un instrument de bază al algebrei liniare și, ca atare, are numeroase aplicații în multe domenii ale matematicii, precum și în matematica aplicată, statistică, fizică, economie și inginerie. Calculul produselor matriciale este o operație centrală în toate aplicațiile algebrei liniare. Produsul matricelor A și B este notat AB. Dacă A este o matrice $m \times n$, iar B este o matrice $n \times p$, atunci:

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{m1} & a_{m2} & ... & a_{mn} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & ... & b_{1p} \\ b_{21} & b_{22} & ... & b_{2p} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{np} \end{pmatrix},$$

$$C = \begin{pmatrix} c_{11} & c_{12} & ... & c_{1p} \\ c_{21} & c_{22} & ... & c_{2p} \\ ... & ... & ... & ... \\ c_{m1} & c_{m2} & ... & c_{mp} \end{pmatrix},$$

unde C este produsul matriceal $C = AB$, fiind definit ca o matrice de tip $m \times p$. Astfel, fiecare element $c_{ij}, i = \overline{1,m}, j = \overline{1,p}$, se va scrie:

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj} \Rightarrow$$

$$C = \begin{pmatrix} a_{11}b_{11} + ... + a_{1n}b_{n1} & a_{11}b_{12} + ... + a_{1n}b_{n2} & a_{11}b_{1p} + ... + a_{1n}b_{np} \\ a_{21}b_{11} + ... + a_{2n}b_{n1} & a_{21}b_{12} + ... + a_{2n}b_{n2} & a_{21}b_{1p} + ... + a_{2n}b_{np} \\ ... & ... & ... \\ a_{m1}b_{11} + ... + a_{mn}b_{n1} & a_{m1}b_{12} + ... + a_{mn}b_{n2} & a_{m1}b_{1p} + ... + a_{mn}b_{np} \end{pmatrix}.$$

3.2 Analiza complexității timp și spațiu al algoritmului clasic

Cel mai simplu mod de a calcula produsul a două matrice n × n, A și B, este să calculăm expresiile aritmetice care rezultă din definiția înmulțirii matricelor. În pseudocod:

inițializăm matricele A și B, ambele fiind de dimensiune $n \times n$
  pentru $i \leftarrow 1$ până la n:
    pentru $j \leftarrow 1$ până la n:
      pentru $k \leftarrow 1$ până la n:
        $C[i][j] = C[i][j] + A[i][k] * B[k][j]$
  afișăm matricea C

Trivial: Spațiul ocupat este $O(n^2)$.
Analiza algoritmului: Adunările și înmulțirile sunt executate doar o dată în cel mai adânc nivel de buclă. Notăm $M(n)$ și $A(n)$ numărul total de operații de înmulțire, respectiv adunare, efectuate în total. Astfel:

$$M(n), A(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 1 = n^3,$$
$$T(n) = t_m \cdot M(n) + t_a \cdot A(n) = (t_m + t_n) \cdot n^3, \text{ unde}$$
$$t_m = \text{timpul de executare a unei înmulțiri},$$
$$t_a = \text{timpul de executare a unei adunări}.$$

În concluzie, complexitatea înmulțirii a două matrice de dimensiune $n \times n$ este de ordinul $O(n^3)$. Acest lucru se datorează faptului că numărul total de operații de înmulțire crește cubic odată cu dimensiunea matricei, fiind influențat de numărul de elemente din matrice n^2 și de faptul că înmulțirile sunt efectuate într-un triplet de bucle care se execută de n ori fiecare.

4 Metode de optimizare a înmulțirii matricelor

4.1 Metoda Strassen pentru înmulțirea matricelor

Algoritmul Strassen, numit după Volker Strassen, este un algoritm pentru înmulțirea matricelor. Este mai rapid decât algoritmul standard pentru înmulțirea matricelor în cazul matricelor mari, având o complexitate asimptotică mai bună, deși algoritmul naiv este adesea mai eficient pentru matrici mai mici. Algoritmul Strassen este mai lent decât cei mai rapizi algoritmi cunoscuți pentru matrici extrem de mari, dar acești algoritmi de tip galactic nu sunt utili în practică. Volker Strassen a publicat acest algoritm pentru prima dată în 1969 și, astfel, a demonstrat că algoritmul general de înmulțire a matricelor n^3 nu era optim. Publicarea algoritmului Strassen a dus la mai multe cercetări despre înmulțirea matricelor, care au condus atât la limite inferioare asimptotice, cât și la limite superioare computaționale îmbunătățite.
Observație: În algoritmul prezentat, matricele A, B și $C \in M_{2^n \times 2^n}(\mathbb{C}), \forall n \in \mathbb{N}$. Dacă matricele A și B nu sunt de tipul $2^n \times 2^n$, rândurile și coloanele "lipsă" pot fi completate cu zero-uri pentru a obține matrice cu dimensiuni puteri ale lui doi.

Inițial, din obișnuință, calculăm produsul a două matrici de dimensiune 2×2 astfel:

$$A \times B = C \Leftrightarrow$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \Rightarrow$$

$$\begin{aligned} c_{11} &= a_{11} \times b_{11} + a_{12} \times b_{21} \\ c_{12} &= a_{11} \times b_{12} + a_{12} \times b_{22} \\ c_{21} &= a_{21} \times b_{11} + a_{22} \times b_{21} \\ c_{22} &= a_{21} \times b_{12} + a_{22} \times b_{22} \end{aligned}$$

Se poate observa faptul că numărul total de operații este de 8 înmulțiri, respectiv 4 adunări. Algoritmul Strassen a găsit o cale de a rezolva produsul celor 2 matrici cu doar 7 înmulțiri, în loc de 8 și 18 adunări / scăderi. Acesta își definește următoarele valori:

$$\begin{aligned} P_1 &= (a_{11} + a_{22}) \times (b_{11} + b_{22}) \\ P_2 &= (a_{21} + a_{22}) \times b_{11} \end{aligned}$$

$$P_3 = a_{11} \times (b_{12} - b_{22})$$
$$P_4 = a_{22} \times (b_{21} - b_{11})$$
$$P_5 = (a_{11} + a_{12}) \times b_{22}$$
$$P_6 = (a_{21} - a_{11}) \times (b_{11} + b_{12})$$
$$P_7 = (a_{12} - a_{22}) \times (b_{21} + b_{22})$$
$$C_{11} = P_1 + P_4 - P_5 + P_7$$
$$C_{12} = P_3 + P_5$$
$$C_{21} = P_2 + P_4$$
$$C_{22} = P_1 - P_2 + P_3 + P_6$$

Folosind o strategie recursivă de tip "Divide Et Impera" (presupunem că $a_{11}, ..., a_{22}, b_{11}, ..., b_{22}$ sunt niște matrici de dimensiune $\frac{n}{2} \times \frac{n}{2}$) permite înmulțirea matricelor pentru dimensiunea $N = 2^n$ folosindu-se doar $7^N = 7^{\log_2(n)} = n^{\log_2(7)} = O(n^{2.81})$ operații de multiplicare.

Pseudocod:
Strassen(A, B):
dacă n = 1, atunci $A \times B$
altfel:
împartim $a_{11}, ..., a_{22}, b_{11}, ..., b_{22}, c_{11}, ..., c_{22}$ în submatrici egale
$P_1 \leftarrow Strassen(a_{11}, b_{12} - b_{22})$
$P_2 \leftarrow Strassen(a_{11} + a_{12}, b_{22})$
$P_3 \leftarrow Strassen(a_{21} + a_{22}, b_{11})$
$P_4 \leftarrow Strassen(a_{22}, b_{21} - b_{11})$
$P_5 \leftarrow Strassen(a_{11} + a_{22}, b_{11} + b_{22})$
$P_6 \leftarrow Strassen(a_{12} - a_{22}, b_{21} + b_{22})$
$P_7 \leftarrow Strassen(a_{11} - a_{21}, b_{11} + b_{12})$
$c_{11} = P_5 + P_4 - P_2 + P_6$
$c_{12} = P_1 + P_2$
$c_{21} = P_3 + P_4$
$c_{22} = P_1 + P_5 - P_3 - P_7$
afișează C
sfârșit subprogram

4.2 Analiza complexității timp și spațiu al noului algoritm

Sunt șapte apeluri recursive, iar adunările și scăderile durează timpul cn^2 pentru o constantă c. Prin urmare, timpul de execuție $T(n)$ satisface:

$$T(n) \leq 7T(\tfrac{n}{2}) + cn^2$$
$$T(1) = 1$$

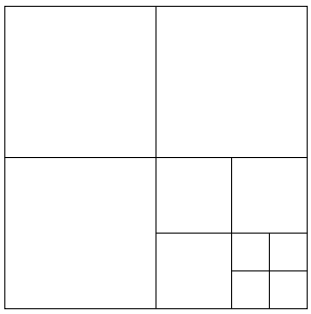
Observație: În primul rând, chiar dacă este asimptotic mai rapid decât algoritmul naiv, Algoritmul lui Strassen nu devine mai rapid decât algoritmul naiv decât atunci când dimensiunea matricelor este de ordinul a 10.000. În loc să recursăm până la matrice de dimensiune 1×1, putem rula pur și simplu algoritmul naiv o dată când matricile devin suficient de mici pentru a obține un

algoritm mai rapid.

Fiind mai atenți la modul în care folosim spațiul, putem rula, de asemenea, Algoritmul lui Strassen în $O(n^2)$. În loc să calculăm fiecare P_i în propriul său spațiu, putem folosi același spațiu pentru fiecare dintre aceste apeluri recursive. Pornim cu $3n^2$ spațiu pentru cele două matrici de intrare și matricea de ieșire. Apoi alocăm $3(n/2)^2$ spațiu pentru apelul recursiv pentru a calcula P_1. Odată ce P_1 este returnat, îl adăugăm în colțul din dreapta sus și dreapta jos al matricei de ieșire. În același spațiu de $3(n/2)^2$, calculăm P_2, și apoi îl adăugăm în părțile din stânga sus și dreapta sus ale matricei de ieșire. Continuăm în același mod pentru restul de P_i-uri.

Fie W(n) cantitatea de memorie necesară pentru a înmulți două matrice n × n. Din discuția de mai sus, vedem că:

$$W(n) = 3n^2 + W(n/2)$$



Cele trei pătrate de dimensiune $n \times n$ reprezintă spațiul necesar pentru apelul de nivel superior, iar cele trei pătrate de dimensiune $n/2 \times n/2$ reprezintă spațiul necesar pentru primul nivel de recursivitate, etc.

5 Punctul de plecare al eficienței

5.1 Unde s-a ajuns în prezent

Începând cu ianuarie 2024, cea mai bună limită asimptotică pentru complexitatea algoritmului de înmulțire a matricelor este $O(n^{2.371552})$. Cu toate acestea, acesta și alte îmbunătățiri similare aduse algoritmului lui Strassen nu sunt folosite în practică, deoarece sunt algoritmi galactici: coeficientul constant ascuns de notația $O(n^\omega)$ este atât de mare încât sunt demne de luat în considerare doar pentru matricele prea mari pentru a fi manipulate pe calculatoarele de azi. Totuși, este important să li se acorde atenție și să fie studiate în profunzime pentru a reuși în a găsi un răspuns la o întrebare care încă nu are răspuns: **"Care este cel mai rapid algoritm pentru înmulțirea matricelor?"**

Linia temporală a exponentului de înmulțire a matricelor

An	Limită superioară pentru ω	Autor
1969	2.8074	Strassen
1978	2.796	Pan
1979	2.780	Bini, Capovani, Romani
1981	2.522	Schönhage
1981	2.517	Romani
1981	2.496	Coppersmith, Winograd
1986	2.479	Strassen
1990	2.3775	Coppersmith, Winograd
2010	2.3737	Stothers
2012	2.3729	Williams
2014	2.3728639	Le Gall
2020	2.3728596	Alman, Williams
2022	2.371866	Duan, Wu, Zhou
2024	2.371552	Williams, Xu, Xu, Zhou

5.2 Învățare prin reinforțare asupra înmulțirii matricelor

În subsecțiunea anterioară, toți algoritmii au fost realizați de către cercetători cunoscuți asupra acestui domeniu. În anul 2022, a fost publicat un articol despre descoperirea a unor noi modalități de înmulțire a matricelor cu ajutorul inteligenței artificale. Pentru a asigura o înțelegere adecvată a conținutului acestui articol, vom investiga următoarele întrebări:

5.2.1 Cine a reușit?

Echipa celor de la Deep Mind au creat un agent pe nume **Alpha Tensor** care este antrenat să joace un joc single-player în care obiectivul este găsirea descompunerilor tensoriale într-un spațiu finit de factori. În matematică, un tensor este un obiect geometric care asociază într-o manieră multi-liniară vectori geometrici, scalari și alți tensori cu un tensor rezultat. Exemple uzuale de tensori:

Scalar

$$\begin{bmatrix} \alpha_{11} \end{bmatrix}$$

Vector

$$\begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} \end{bmatrix}$$

Matrice

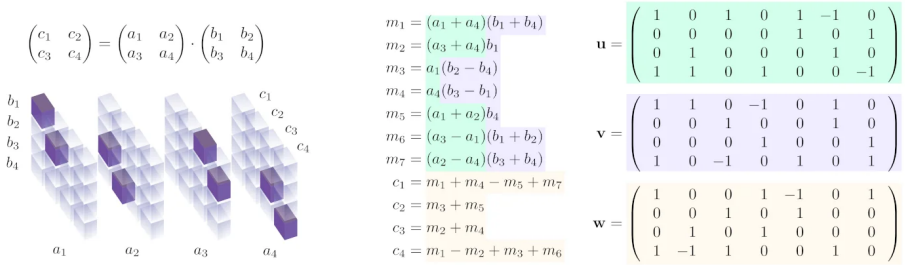
$$\begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

Tensor 3D

$$\left[\left[\begin{bmatrix}\alpha_{11} & \alpha_{12} & \alpha_{13}\\ \alpha_{21} & \alpha_{22} & \alpha_{23}\\ \alpha_{31} & \alpha_{32} & \alpha_{33}\end{bmatrix}\begin{bmatrix}\beta_{11} & \beta_{12} & \beta_{13}\\ \beta_{21} & \beta_{22} & \beta_{23}\\ \beta_{31} & \beta_{32} & \beta_{33}\end{bmatrix}\begin{bmatrix}\gamma_{11} & \gamma_{12} & \gamma_{13}\\ \gamma_{21} & \gamma_{22} & \gamma_{23}\\ \gamma_{31} & \gamma_{32} & \gamma_{33}\end{bmatrix}\right]\right]$$

5.2.2 Cum au reușit?

Cum înmulțirea matricelor este $(A, B) \to AB$ este o operație biliniară, poate fi reprezentată precum un tensor 3D:



Explicații: u, v, w sunt 2 matrici cu 4 linii datorită faptului că sunt 4 elemente în matricea A, B respectiv C și 7 coloane, unde fiecare coloană reprezintă un vector format doar din valorile -1, 0 și 1 (fiecare vector este notat cu $u^{(k)}, v^{(l)}, w^{(m)}$, unde $k, l, m = \overline{1,7}$). Fiecare vector poate fi gândit în felul următor:

-1 $\to$ elementul respectiv este folosit cu minus
0 $\to$ elementul respectiv nu este folosit deloc
1 $\to$ elementul respectiv este folosit cu plus

$$\begin{pmatrix}a_1\\ a_2\\ a_3\\ a_4\end{pmatrix} or \begin{pmatrix}b_1\\ b_2\\ b_3\\ b_4\end{pmatrix} or \begin{pmatrix}c_1\\ c_2\\ c_3\\ c_4\end{pmatrix} = \begin{pmatrix}1\\ 0\\ 0\\ 1\end{pmatrix}$$

Notăm T_n (în acest caz $T_n = T_{n,n,n}$) tensorul care descrie operația de înmulțire. În general, este de dimensiune $n^2 \times n^2 \times n^2$. Prin descompunerea lui T_n în tensori de rang 1, ne referim la faptul că: $T_n = \sum_{r=1}^{R} u^{(r)} \bigotimes v^{(r)} \bigotimes w^{(r)}$, unde notația "$\bigotimes$" reprezintă produsul tensorial.

Exemplu de produs tensorial:

$$u^{(1)} \bigotimes v^{(1)} \bigotimes w^{(1)} = \begin{bmatrix}1\\0\\0\\1\end{bmatrix} \bigotimes \begin{bmatrix}1\\0\\0\\1\end{bmatrix} \bigotimes \begin{bmatrix}1\\0\\0\\1\end{bmatrix} =$$

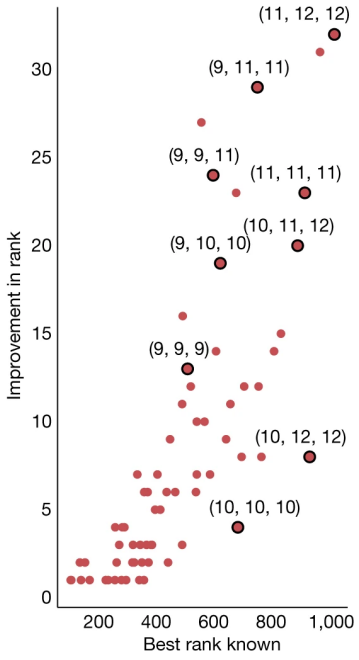
$$\begin{bmatrix}1&0&0&1\\0&0&0&0\\0&0&0&0\\1&0&0&1\end{bmatrix} \bigotimes \begin{bmatrix}1\\0\\0\\1\end{bmatrix} = \left[\begin{bmatrix}1&0&0&1\\0&0&0&0\\0&0&0&0\\1&0&0&1\end{bmatrix}\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\begin{bmatrix}1&0&0&1\\0&0&0&0\\0&0&0&0\\1&0&0&1\end{bmatrix}\right]$$

Dacă jocul ar fi avut ca temă "Optimizează produsul celor două matrici pentru a folosi cât mai puține operații de înmulțire", ar fi fost una vagă. În schimb, ar putea fi definită și specifică pentru agent în felul următor: "Descompune un tensor tridimensional într-un număr minim de tensori de rang 1". Astfel, descompunerea lui T_n în R termeni de rang 1, determină un algoritm pentru produsul celor două matrici folosind doar R operații de înmulțire între termenii lor. Complexitatea este de $O(N^{\log_n(R)})$, unde $N \times N$ reprezintă dimensiunea matricelor inițiale, iar n reprezintă dimensiunea matricei pentru care s-a găsit un algoritm eficient.

5.2.3 Ce au reușit?

Una dintre cele mai mari reușite a fost să îmbunătățească algoritmul clasic de înmulțire al lui Volker Strassen pentru matricele de dimensiuni 4×4 în $\mathbb{Z}_2$ folosind doar 47 de înmulțiri, în loc de 49. Mai mult, a descoperit algoritmi eficienți de multiplicare în aritmetica standard pentru dimensiuni mai mari.

Size ⟨n, m, p⟩	Best method known	Best rank known	AlphaTensor rank Modular	AlphaTensor rank Standard
⟨2, 2, 2⟩	(Strassen, 1969)[2]	7	7	7
⟨3, 3, 3⟩	(Laderman, 1976)[15]	23	23	23
⟨4, 4, 4⟩	(Strassen, 1969)[2] ⟨2, 2, 2⟩ ⊗ ⟨2, 2, 2⟩	49	47	49
⟨5, 5, 5⟩	⟨3, 5, 5⟩ + ⟨2, 5, 5⟩	98	96	98
⟨2, 2, 3⟩	⟨2, 2, 2⟩ + ⟨2, 2, 1⟩	11	11	11
⟨2, 2, 4⟩	⟨2, 2, 2⟩ + ⟨2, 2, 2⟩	14	14	14
⟨2, 2, 5⟩	⟨2, 2, 2⟩ + ⟨2, 2, 3⟩	18	18	18
⟨2, 3, 3⟩	(Hopcroft and Kerr, 1971)[16]	15	15	15
⟨2, 3, 4⟩	(Hopcroft and Kerr, 1971)[16]	20	20	20
⟨2, 3, 5⟩	(Hopcroft and Kerr, 1971)[16]	25	25	25
⟨2, 4, 4⟩	(Hopcroft and Kerr, 1971)[16]	26	26	26
⟨2, 4, 5⟩	(Hopcroft and Kerr, 1971)[16]	33	33	33
⟨2, 5, 5⟩	(Hopcroft and Kerr, 1971)[16]	40	40	40
⟨3, 3, 4⟩	(Smirnov, 2013)[18]	29	29	29
⟨3, 3, 5⟩	(Smirnov, 2013)[18]	36	36	36
⟨3, 4, 4⟩	(Smirnov, 2013)[18]	38	38	38
⟨3, 4, 5⟩	(Smirnov, 2013)[18]	48	47	47
⟨3, 5, 5⟩	(Sedoglavic and Smirnov, 2021)[19]	58	58	58
⟨4, 4, 5⟩	⟨4, 4, 2⟩ + ⟨4, 4, 3⟩	64	63	63
⟨4, 5, 5⟩	⟨2, 5, 5⟩ ⊗ ⟨2, 1, 1⟩	80	76	76



6 Bibliografie

1. **Divide-and-Conquer algorithms for matrix multiplication**
2. **The Coppersmith-Winograd Matrix Multiplication Algorithm**
3. **Introduction and Strassen's Algorithm**
4. **Discovering faster matrix multiplication algorithms with reinforcement learning**
5. **How AI Discovered a Faster Matrix Multiplication Algorithm**
6. **Computational complexity of matrix multiplication**