

Laborator 6

Aplicația 1: polinoame (egale?)

Să sănătățim două polinoame de grad (cel mult) d :

$$F(x) = \prod_{i=1}^d (x - a_i)$$

$$G(x) = \sum_{i=0}^d b_i x^i$$

forma canonica

Propunem un algoritm randomizat care să verifice dacă $F(x) = G(x)$, dar fără a-l aduce pe F la forma canonica. Urmărim o eficiență cît mai mare (i.e. P | algoritmul greseste) și reducerea volumului de calcul.

Procedere:

Partea algoritmului propus:

- alegem un număr r uniform în mulțimea $\{1, \dots, 100 \cdot d\}$
 - ↑ analogul teoretic este echipepartiția pe $\Omega = \{1, \dots, 100 \cdot d\}$.
- dacă $F(r) \neq G(r)$, atunci se returnează că cele două polinoame NU sunt egale
- dacă $F(r) = G(r)$, atunci se afișează că cele două polinoame sunt identice.

Exemplu-problemă: $\begin{cases} F(x) = (x+1)^2 \\ G(x) = 2x+1 \end{cases}$ $F \neq G$, dar $F(0) = 1 = G(0)$

Obțăm cu E = evenimentul că algoritmul greșeste. Vrem să evaluăm probabilitatea acestuia.

Q: Când se realizează acest eveniment E ?

A: Atunci când $F(r) = G(r)$, totul e clar; polinoame sunt rigurătoare și concluzia e corectă. Algoritmul poate greși doar dacă cele două polinoame H sunt egale, dar $F(r) = G(r)$ [! pt. valoarea r selectată]. Asta înseamnă că valoarea r este / trebuie să fie rădăcine a polinomului $H(r) = F(r) - G(r)$.

Deoarece $\text{grad}(F) \leq d$ și $\text{grad}(G) \leq d$, avem că $\text{grad}(H) \leq d$. Apoi din Teorema fundamentală a algebrei obținem că H nu poate avea mai mult de d rădăcini. În alte cuvinte, algoritmul greșete cu probabilitatea:

$$P(E) = \frac{\#\text{casuri favorabile}}{\#\text{casuri posibile}} \stackrel{!}{=} \frac{d}{100 \cdot d} = \boxed{\frac{1}{100}}$$

Obs: $P(\text{algoritmul grește}) \leq \frac{1}{100} \Rightarrow P(\text{algoritmul nu grește})$

$$\boxed{P(E)}$$

$$P(E^c) = 1 - P(E) \geq 1 - \frac{1}{100}$$

*IMBUNATATIRE 1: mărim spațiul algoritmilor de la $\{1, \dots, 100 \cdot d\}$ la $\{1, \dots, 10^3 \cdot d\}$, de exemplu. Atunci avem că:

$$P(E) \leq \frac{d}{10^3 \cdot d} = \boxed{\frac{1}{1000}}.$$

*IMBUNATATIRE 2: repetăm evaluarea polinoamelor de m ori și emitem o concluzie abia la final.

VARIANTA 2a): „replace = TRUE” (i.e. nu ținem cont de nr. aleatoare și selectate anterior)

Aveam evenimentele $(E_i)_{i=1, \dots, m}$, unde E_i = evenimentul că algoritmul găsește la a i -a iteratie.

Evenimentele $(E_i)_{i=1, \dots, m}$ sunt independente și $P(E_i) \leq \frac{1}{100}$, $\forall i=1, \dots, m$. Mai departe, rezultă că:

$$P(E_1 \cap \dots \cap E_m) = \prod_{i=1}^m P(E_i) \leq \left(\frac{1}{100}\right)^m$$

VARIANTA 2b): „replace = FALSE” (i.e. valourile și selectate anterior sunt omise)

În acest caz, evenimentele $(E_i)_{i=1, \dots, m}$ nu mai sunt independente. În schimb, știm că:

$$P(E_i | E_{i-1}, \dots, E_1) \leq \frac{d - (i-1)}{100d - (i-1)}$$

În plus, obținem aici că:

$$\text{ex: } P(E_2 | E_1) \leq \frac{d-1}{100d-1}.$$

$$P(E_1 \cap \dots \cap E_m) = P(E_1) \cdot P(E_2 | E_1) \cdot P(E_3 | E_1 \cap E_2) \cdot \dots \cdot P(E_m | E_1 \cap \dots \cap E_{m-1})$$

$$\leq \prod_{i=1}^m \frac{d - (i-1)}{100d - (i-1)} \stackrel{\text{?}}{\leq} \prod_{i=1}^m \frac{1}{100} = \left(\frac{1}{100}\right)^m$$

A se vedea scrisă R pt. implementare. ■

Aplicația 2: produsul a două matrice

Se dau 3 matrice pătratice de dimensiune $n \times n$: A, B, C . Pentru simplitate, vom considera că elementele matricelor pot lua doar valorile 0 și 1, iar operațiile se fac modulo 2.

Construiești un algoritm randomizat, care să verifice egalitatea: $A \cdot B = C \pmod{2}$.

Programare:

Algoritmul de bază (al lui Freivalds) are 3 pași, anume:

PAS 1: Alegem uniform un vector $r = (r_1, \dots, r_n) \in \{0,1\}^n$. Altfel spus, fiecare componentă r_i este aleasă uniform din multimea $\{0,1\}$ și independent de celelalte componente.

PAS 2: Calculăm $y = A \cdot (B \cdot r)$ și $z = C \cdot r$.

PAS 3: Dacă $y \neq z$, se afișează că rezultatul multiplicării este gresit. Altfel, dacă $y = z$, se returnează mesajul că înmulțirea este corectă.

! Algoritmul găsește atunci când $ABr = Cr$, dar — de fapt — $AB \neq C$.

• De situația acum în contextul în care $AB \neq C$. Notăm că $D = AB - C$. Este clar atunci că $D \neq 0_{m \times n}$ și putem presupune că $d_{1,1} \neq 0$.

• Mai departe, dacă $AB \cdot r = C \cdot r$, atunci:

$$\Leftrightarrow AB \cdot r - Cr = 0_m$$

$$\Leftrightarrow (AB - C)r = 0$$

$$\Leftrightarrow D \cdot r = 0$$

$$\Rightarrow \sum_{i=1}^n d_{1,i} \cdot r_i = 0$$

$$\left(\begin{array}{c} \textcircled{1} \\ \vdots \end{array} \right) \left(\begin{array}{c} r \\ \vdots \end{array} \right) = \left(\begin{array}{c} \boxed{0} \\ \vdots \\ 0 \end{array} \right)$$

$$\Leftrightarrow d_{1,1} \cdot r_1 + \sum_{i=2}^n d_{1,i} \cdot r_i = 0$$

$$\Leftrightarrow d_{1,1} \cdot r_1 = - \sum_{i=2}^n d_{1,i} \cdot r_i$$

$$d_{1,1} \neq 0$$

$$\Leftrightarrow$$

$$r_1 = - \frac{\sum_{i=2}^n d_{1,i} \cdot r_i}{d_{1,1}}$$

$$\in \{0, 1\}$$

• În concluzie,

$$P(\text{algoritmul greșeste}) = P(AB \cdot r \neq C \cdot r)$$

pt A, B, C fixate
cu $AB \neq C$

$$= \sum_{x_2, \dots, x_n \in \{0, 1\}} P(\{AB \cdot r = C \cdot r\} \wedge \{(r_2, \dots, r_n) = (x_2, \dots, x_n)\})$$

$$\leq \sum_{x_2, \dots, x_n \in \{0, 1\}} P\left(r_1 = - \frac{\sum_{i=2}^n d_{1,i} \cdot r_i}{d_{1,1}} \wedge (r_2, \dots, r_n) = (x_2, \dots, x_n)\right)$$

$$\Rightarrow P(\text{algoritmul greseste}) \stackrel{\text{(ind)}}{\leq} \sum_{x_2, \dots, x_n \in \{0,1\}} P\left(\lambda_1 = -\frac{\sum_{i=2}^n d_{1,i} \cdot \lambda_i}{d_{1,1}}\right) \cdot P((\lambda_2, \dots, \lambda_n) = (x_2, \dots, x_n))$$

$$= \sum_{(x_2, \dots, x_n) \in \{0,1\}^{n-1}} \frac{1}{2} \cdot P((\lambda_2, \dots, \lambda_n) = (x_2, \dots, x_n))$$

$$= \frac{1}{2} \cdot \sum_{(x_2, \dots, x_n) \in \{0,1\}^{n-1}} P((\lambda_2, \dots, \lambda_n) = (x_2, \dots, x_n))$$

$$= \frac{1}{2} \cdot 1 = \boxed{\frac{1}{2}}$$

! Pentru a îmbunătăți eroarea algoritmului, putem să repetăm de k ori procedeul. În acest, probabilitatea de eroare devine $2^{-k} = \left(\frac{1}{2}\right)^k$.

A se vedea scrisă R pt. implementare. ■