RANDOMIZÁCIÓ ÉS PROGRAMTESZTELÉS

RANDOMIZATION AND PROGRAM TESTING

dr. Nagy Ferenc

ABSTRACT

The implementation of an algorithm is the product of the programmer. It may be correct or it may be incorrect. Tests are used to make decision to accept or to reject the software. This paper describes a statistical way of decision making. It extends the collection of the methods of the program testing. The main idea is the concept of the information geometry. An example demonstrates the usefulness of this idea.

1. BEVEZETÉS

Ebben a cikkben a számítógépes program programhelyesség vizsgálatának kérdéséhez járulunk hozzá, a véletlen szerepének feszegetésével. A vázolt módszer sztochasztikus döntést eredményez, ennek megfelelően tévedhet is. A kérdés csupán a tévedés gyakoriságának nagysága. Nem oldjuk meg a teljes programhelyesség vizsgálat problémáját, – magáról a problémáról [1]-ben olvashatunk – csupán helyenként alkalmazható hasznos tesztek konstruálására adunk módszert, amelyekben részint elkerülhető a teszt konstrukciójának és az input adatok kiválasztásának a problémája.

Egy számítógépes program egy absztrakt algoritmus implementációja. Tulajdonképpen az implementációt nevezzük programnak. Maga az algoritmus is és az implementáció is egy függvény, amely az $x \in X$ input adathoz, a $z \in Z$ output elemet (adatot) rendeli hozzá. Az input adat is és az output adat is lehet bonyolult összetételű. Ha A jelöli az algoritmust, P pedig a programot, akkor az input leképezése az outputra az $A: X \to Z$ és a $P: X \to Z$ módon írható le. A program, az implementáció megírása, elkészítése a programspecifikáció realizálását jelenti a választott programozási nyelv szintaxisának megfelelően, amikor realizáljuk az input és az output elemek egymáshoz rendelését. Kérdéses azonban, hogy a megírt program valóban a valódi, helyes specifikációt realizálja-e vagy sem. A célunk kezdetben az, hogy eldönthessük, hogy hibás-e a program, a miért és a hiba helye, jellege kezdetben kevésbé lényeges. A hiba helyének és milyenségének a megállapítása sokkal bonyolultabb feladat, ami messzire vezet, és most nem célunk. Tehát a programozó elkészítette a $\widetilde{P}$ programot, amelyre $\widetilde{P}: X \to Z$, és nekünk el kell döntenünk, hogy $\widetilde{P} = P$, vagy $\widetilde{P} \neq P$ áll-e fenn, azaz a két leképezés, az elkészült és az elképzelt ekvivalens-e, vagy sem. Ha nem ekvivalensek, akkor legalább egy helyen a két leképezés eltérő outputot eredményez. Általában igen gondosan megtervezett tesztekkel próbáljuk kideríteni, hogy a program minden ágon a specifikációnak megfelelően fut-e. Ezeket nem könnyű megtervezni, a szükséges input adatokat megválasztani, a mennyiségüket megbecsülni. Sok, különböző tesztet kell előkészíteni, amelyek mintegy letapogatják az elkészült programot, előre ismerve, hogy a megtervezett inputra mi kell, hogy legyen a program outputja. Egy gyors teszt segíthet legalább azt a gyanúnkat megerősíteni, hogy a program hibás, ha valóban hibás. Ezen célból javasoljuk a teszt randomizálását, azaz véletlenszerűen választott inputokkal programfuttatásokat végezni. A mai multi taszkos, multiprocesszoros számítógépekkel ez általában nem okoz különösebben gondot. A randomizációval kapcsolatban, messze nem a teljesség igényével, megemlítjük a [2], [4], [5], [8] munkákat. Az inputok X terén (halmazán) egy valószínűségi mértéket, valószínűség eloszlást definiálunk. Ennek megválasztása tőlünk függ. Ez egy ξ véletlen változó megválasztását jelenti. Ez lesz a P és $\widetilde{P}$ programok inputja. Az előbbi a helyes, az utóbbi a helytelen realizáció esete. A programok erre az inputra válaszként, outputként a ζ illetve a $\widetilde{\zeta}$ véletlen változót adják a Z output téren. Elviekben az A algoritmus ismeretében ζ eloszlása pontosan meghatározható. Tehát a P programot ténylegesen nem kell elkészítenünk. A $\widetilde{P}$ program esetében pedig $\widetilde{\zeta}$-ra nagyszámú megfigyelés, uniformizált teszt végezhető, amely lehetőséget ad eloszlásának becslésére. Ezekben a tesztekben nem kell megtervezni az inputokat, mivel azok véletlenszerűen kerülnek előállításra, generálásra, és nem kell ismerni előre a rájuk adandó outputokat sem. A problémánk ezáltal arra redukálódik, hogy eldöntsük, hogy a kétféle eloszlás eltér-e egymástól, vagy sem. Ha eltérnek, akkor

bizonyosan helytelen az implementáció. Erre a célra hipotézis vizsgálatot végezhetünk.

2. A MÓDSZER ÉS AZ ESZKÖZ

Egy sor becslési és dimenzió probléma elkerülhető, ha az információelméleti entrópia, inakkurancia és diszkrimináló információ (I-divergencia, Kullback-divergencia) fogalmakat hívjuk segítségül [3], [7]. Emlékeztetőül ezeket a fogalmakat az alábbi várható érték formulákkal definiáljuk. A formulákban *E* a várható érték képzésének a jele. Legyenek ξ, ξ_1 és ξ_2 véletlen változók, akár többdimenziósak, rendre az $f(x), f_1(x)$ and $f_2(x)$ sűrűségfüggvényekkel. Az entrópia (*H*), az inakkurancia (*T*) és a diszkrimináló információ (*D*):

$$H(\xi) = E_{\xi}\left(\log\frac{1}{f(x)}\right), \tag{2.1}$$

$$T(\xi_1\|\xi_2) = E_{\xi_1}\left(\log\frac{1}{f_2(x)}\right), \tag{2.2}$$

$$D(\xi_1\|\xi_2) = E_{\xi_1}\left(\log\frac{f_1(x)}{f_2(x)}\right). \tag{2.3}$$

A lényegen nem változtat az a tény, hogy folytonos véletlen változóra adtuk meg a formulákat, mert diszkrét esetben is érvényesek az elmondásra kerülő dolgok. A diszkrimináló információ nemnegatív, és akkor és csak akkor zérus, ha a két sűrűségfüggvény majdnem mindenütt (azaz legfeljebb zérusmértékű halmaz kivételével) megegyezik [6]. Egyfajta eltérést jelez a két eloszlás között. Általában igaz, hogy $D(\xi_1\|\xi_2) \neq D(\xi_2\|\xi_1)$. Az is teljesül, hogy

$$D(\xi_1\|\xi_2) = T(\xi_1\|\xi_2) - H(\xi_1). \tag{2.4}$$

Rögzített ξ_2-re (2.4)-ből következik, hogy $T(\xi_1\|\xi_2)$ értéke $\xi_1 = \xi_2$-nél éppen $H(\xi_2)$. A ξ_1, ξ_2 véletlen változók elnevezése (2.2) és (2.3)-ban rendre *a posteriori* és *a priori* változók.

Az A algoritmus véletlen outputot eredményez egy véletlen inputból a randomizáció révén. Ha az algoritmus implementációja hibás, akkor ennek az állapotnak az indikátora lehet a megfigyelt output eloszlás eltérése a várttól. Sajnos az eredmény függ az input eloszlástól, az algoritmustól és az implementációtól, azaz a hibától. Vannak hibák, amelyek nem okoznak eltérést az output eloszlásban. Például szimmetrikus output esetén egy esetleges előjelhiba rejtve maradhat. A rejtett hibákra a 4. fejezetben még visszatérünk. Szerencsére az input eloszlást mi választhatjuk meg, így az output is befolyásolható. Ugyanakkor az output eltérése a várttól mindig a hiba meglétét jelzi. A várt output eloszlást a priorinak, az implementációból kapottat pedig a posteriorinak választjuk Az utóbbit a tesztek révén becsülhetjük meg, ami nagyméretű véletlen generált minták (nagyszámú teszt output) esetén várhatóan megbízható eredményt ad. Felhasználva a generált véletlen tesztek által szolgáltatott outputok empirikus mintaeloszlását, amelyet a posteriori eloszlásnak tekintünk, mint az igazi a posteriori eloszlás becslését, megkísérelhetjük az eltérés kiszámítását. Ezt azonban nem érdemes a (2.4) alapján elvégezni, mivel az empirikus eloszlások esetén a formulában a becsült entrópia tag a végtelenbe fut el, ha a mintaméret minden határon túl nő. Szeretnénk ugyanis a méret növekedtével egyre pontosabb, megbízhatóbb döntésekre jutni. A megoldást az szolgáltatja, hogy ha az implementáció helyes, és az empirikus a posteriori eloszlás a mintaméret növekedésével konvergál az a priorihoz, akkor az empirikus inakkurancia is konvergál az ismert a priori entrópiához. Hibás implementáció esetén ez a konvergencia általában nem áll fenn. Az empirikus inakkurancia, mint az inakkurancia becslése, az alábbi formulával számítható.

$$T(\tilde{\zeta}_k\|\zeta) = E_{\tilde{\zeta}_k}\left(\log\frac{1}{f_{\zeta}(z)}\right) = \sum_{i=1}^{k}\frac{1}{k}\log\frac{1}{f_{\zeta}(z_i)}. \tag{2.5}$$

Megjegyezzük, hogy a (2.5) formula a (2.2) formula alkalmazása az empirikus a posteriori eloszlásra, amely diszkrét eloszlás. A formulában z_i az implementációra vonatkozó *k* elemű minta *i.* eleme, az *i.* teszt outputja. Szükségünk van statisztikai döntéshozatalra, hipotézis vizsgálatra, amely eldönti, hogy $T(\tilde{\zeta}_k\|\zeta)$ és $H(\zeta)$ eltérése szignifikáns-e, vagy sem, mert még hibamentes esetben sem várható a becslés és a pontos érték egyezése, tehát csak statisztikai egyezőségre, vagy nem egyezőségre számíthatunk. Ez, és ehhez hasonló megközelítés került alkalmazásra [9], [10]-ben.

3. PÉLDA

Legyen $A \in R^{n\times n}$ mátrix, $x,y \in R^n$ vektor. Legyen egy egyszerű problémánk az $y=Ax$ mátrix-vektor szorzat kiszámítása rögzítettnek feltételezett *A* mátrix, és *n* dimenzió esetére, pontosabban a feladat ennek az implementálása, és az implementáció helyességének ellenőrzése. Tegyük fel, hogy *A* nem szinguláris mátrix. Válasszuk input eloszlásnak a ξ *n*-dimenziós

standard normális véletlen változó esetét. Randomizálást követően a problémánk az $\zeta = A\xi$ esetbe megy át. A ξ sűrűségfüggvénye

$$f_{\xi}(x) = \frac{1}{\sqrt{(2\pi)^n}} \cdot e^{-\frac{1}{2}x^T x}, x \in R^n. \quad (3.1)$$

Jól ismert, hogy ekkor a ζ output eloszlása szintén *n*-dimenziós normális eloszlás, melynek sűrűségfüggvénye

$$f_{\zeta}(z) = \frac{1}{\sqrt{(2\pi)^n \cdot \det(S)}} \cdot e^{-\frac{1}{2}z^T S^{-1} z}, z \in R^n, \quad (3.2)$$

ahol $S = AA^{\mathrm{T}}$ a véletlen output kovariancia mátrixa.

Egy lehetséges helyes implementáció pszeudokódja lehet az alábbi.

```
FOR   i←1     TO n DO
      w←0
      FOR     j←1 TO n DO
              w←w+a_ij·x_j
      y_i←w.
```

„Rontsuk" el ezt az implementációt azáltal, hogy az első és a második sorát felcseréljük. (Tipikus programozási hiba a gyűjtőrekesz nem megfelelő helyen történő inicializálása.) Most a ζ entrópiája

$$H(\zeta) = \int_{R^m} f_{\zeta}(z) \log \frac{1}{f_{\zeta}(z)} dz = \\ = \frac{1}{2}\left[n \cdot \log(2\pi) + \log\det(S) + n\right]. \quad (3.3)$$

Az inakkurancia az empirikus $\tilde{\zeta}_k$ output eloszlás és ζ eloszlása között

$$T(\tilde{\zeta}_k \| \zeta) = \frac{1}{k} \cdot \sum_{i=1}^{k} \log \frac{1}{f_{\zeta}(z_i)} = \\ = \frac{1}{k} \cdot \sum_{i=1}^{k} \log\left(\sqrt{(2\pi)^n \cdot \det(S)} \cdot e^{\frac{1}{2}z_i^T S^{-1} z_i} \right). \quad (3.4)$$

A hipotézis vizsgálat a következők alapján végezhető most. A nagy számok törvénye alapján (3.4) várható értéke, feltételezve a helyes implementációt, $H(\zeta)$. Helytelen implementáció esetén lehet eltérés. Nagy mintaméret esetén alkalmazhatjuk a centrális határeloszlás tételt, amely arra vezet, hogy $T(\tilde{\zeta}_k \| \zeta)$ eloszlása gyakorlatilag normális eloszlású $H(\zeta)$ várható értékkel. Felhasználva a $\log f_{\zeta}(z_i)$ értékeket, becsülni tudjuk $T(\tilde{\zeta}_k \| \zeta)$ szórásnégyzetét az empirikus szórásnégyzettel. Ezután végrehajtunk egy statisztikai *t*-tesztet, amely segít eldönteni, hogy a $T(\tilde{\zeta}_k \| \zeta)$ és $H(\eta)$ közötti eltérés szignifikánsnak tekinthető-e, vagy sem. Bár jelen esetben a *t*-teszt *u*-tesztre is cserélhető, ha a mintaméret elegendően nagy, mivel ilyen szabadsági fokok esetén a két teszt kritikus értékei már gyakorlatilag nem különböznek egy rögzített (mondjuk 1%-os) szignifikancia szinten.

MATLAB programot készítettünk a fentebb leírt probléma tesztelésére és a hipotézis vizsgálatra. Az alábbi táblázatok a döntések eredményeit tartalmazzák. A táblázatokban lévő számérték azon döntések számát mutatja, amelyekben az implementációt helyesnek fogadtuk el.

Helyes implementáció		Tesztek száma			
		10	100	1000	10000
Minta-méret	10	10	99	997	9946
	100	10	100	1000	10000
	1000	10	100	1000	10000
	10000	10	100	1000	10000

Hibás implementáció		Tesztek száma			
		10	100	1000	10000
Minta-méret	10	10	100	1000	9994
	100	2	33	349	3661
	1000	0	0	3	31
	10000	0	0	0	0

Minden teszt esetén rögzítettünk egy mátrixot, és arra végeztük el a randomizált futtatásokat és a hipotézis vizsgálatot. Látható, ahogyan a mintaméret növekedtével a téves döntések százalékos száma rohamosan csökken.

4. REJTETT HIBÁK

Vannak olyan implementációs hibák, amelyek a 2. fejezetben említett nagyon egyszerű eseten túlmutatóan a tesztelésnél rejtve maradnak. Ez arra utal, hogy nem árt az input eloszlást gondosan megválasztani, hogy ezt elkerülhessük. Nem rossz ötlet többféle input eloszlással is próbálkozni.

Tekintsük ugyanis a következő egyszerű problémát. Adott két nemnegatív szám és határozzuk

meg a nagyobbikat. Azaz az algoritmus inputja két nemnegatív szám, az output pedig a kettő közül a nagyobbik (pontosabban a nem kisebbik). Röviden jelölve: meghatározandó $z = \max(x_1, x_2)$. Az implementációjának a pszeudokódja lehet az alábbi.

```
z← x1
IF   z < x2
     THEN z← x2
```

Randomizálva a problémát az átalakul a következővé: $\zeta = \max(\xi_1, \xi_2)$. Természetesnek tűnik a randomizálásnál az input eloszlásának két, egymástól független $\lambda = 1$ paraméterű, exponenciális eloszlást választani. Nem nehéz utána számolni, hogy ebben az esetben a $\zeta = \max(\xi_1, \xi_2)$ eloszlásának sűrűségfüggvénye

$$f_\zeta(z) = \begin{cases} 2e^{-z}\left(1 - e^{-z}\right), & ha\ z \geq 0 \\ 0 & egyébként \end{cases}. \qquad (4.1)$$

Az entrópiát (2.1) alapján rutin integrálással kiszámítva kapjuk, hogy $H(\zeta) = 2 + \log\frac{1}{2}$. Tegyük fel ezek után, hogy a fenti pszeudokódban egy index „elgépelés" történt, az utolsó sorban az x indexe nem kettő, hanem egy lett. Ebben az esetben a z output a $\max(x_1, x_2)$ helyett x_1 lesz. A randomizálás eredményeképpen pedig $\lambda = 1$ paraméterű, exponenciális eloszlású véletlen mennyiséget kapunk, tehát $\tilde{\zeta} = \xi_1$. Kiszámítva a $T(\tilde{\zeta} \| \zeta)$ inakkuranciát, az ugyancsak $2 + \log\frac{1}{2}$-nek bizonyul. Ebből pedig az következik, hogy a randomizálással elvégzett tesztek nem fogják jelezni ezt a hibát. Természetesen, ha megváltoztatjuk az input eloszlást, akkor kaphatunk jelezhető eltérést. Rögzített a priori eloszlásokra nézve azon a posteriori eloszlások, amelyek az a priori entrópiával megegyező inakkuranciát adnak, konstruktívan jellemezhetők. A jellemzés alapján konstruálhatók is.

A bemutatott kutató munka a TÁMOP-4.2.1.B-10/2/KONV-2010-0001 jelű projekt részeként az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

5. IRODALOM

[1] ALAGIĆ S.; ARBIB, M. A.: The Design of Well Structured and Correct Programs, Springer-Verlag, New York Heidelberg Berlin, 1978

[2] Del CORSO G. M.: Estimating an eigenvector by the power method with a random start; SIAM J. Matrix Anal. Appl., 18, No. 4. pp. 913-937, 1997

[3] GRAY R. M.: Entropy and Information Theory, Springer Verlag, New York, Berlin, 1990

[4] JESSUP E. R.: A Statistical Evaluation of Inverse Iteration, NATO ASI Series, Vol. F70, Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, Edited by G. H. Golub and P. Van Dooren, Springer-Verlag Berlin Heidelberg, 1991

[5] KUCZYŃSKI J.; WOZNIAKOWSKI H.: Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start; SIAM J. Matrix Anal. Appl., 13(1992), pp. 1094-1122

[6] KULLBACK S.: Information Theory and Statistics, John Wiley & Sons. Inc. New York, 1959

[7] MATHAI A. M., RATHIE P. N.: Basic Concepts in Information Theory and Statistics. Axiomatic Foundations and Applications, Wiley Eastern Limited, New Delhi, 1974

[8] NAGY F.: Two-sided probability bound estimates in the randomized power method, MicroCAD 2004 Conference, Miskolc, 2004

[9] NAGY F.: Algoritmusok véletlen paraméterei és programok véletlen tesztelése , PhD disszertáció, Miskolci Egyetem, Miskolc 2011.

[10] NAGY F.: Parameter Estimation of the Cauchy Distribution in Information Theory Approach, Journal of Universal Computer Science, vol. 12 No. 9 (2006) pp. 1332-1344.