Iterativni izračun Nashevega ravnotežja v matričnih igrah

Eva Babnik

Maj, 2022

1 Nashevo ravnotežje in vrednost igre v matričnih igrah

Spodobi se, da najprej v nekaj stavkih opišem Nashevo ravnotežje ter vrednost igre v matričnih igrah z ničelno vsoto za dva igralca. Matrično igro z dvema igralcema lahko predstavimo z matriko izplačil $A = (a_{ij})$, kjer prvi igralec izbere eno izmed m vrstic, drugi pa hkrati izbere enega izmed n stolpcev. $A_{i\cdot}$ naj označuje i-to vrstico, $A_{\cdot j}$ pa j - ti stolpec. Če igralca izbereta i-to vrstico in j-ti stolpec, potem drugi igralec plača prvemu igralcu a_{ij}.

Če prvi igralec izbere i-to vrstico z verjetnostjo x_i in drugi izbere j-ti stolpec z verjetostjo y_j, pri čemer velja:

$$x_i \geq 0, \tag{1}$$

$$\sum x_i = 1, \tag{2}$$

$$y_i \geq 0, \tag{3}$$

$$\sum y_i = 1. \tag{4}$$

Potem je pričakovano izplačilo prvemu igralcu $\sum\sum a_{ij}x_iy_j$. Poleg tega velja tudi:

$$\min_j \sum_i a_{ij}x_i \leq \max_i \sum_j a_{ij}y_j.$$

Trditev o minimaksu nam pove, da za neka vektorja verjetnosti $X = (x_1, \cdots, x_m)$ in $Y = (y_1, \cdots, y_n)$ v zgornji enačbi velja enakost. Tak par (X^*, Y^*) predstavlja optimalno strategijo igre. Vrednost igre v pa je definirana kot:

$$v = \min_j \sum_i a_{ij}x_i = \max_i \sum_j a_{ij}y_j.$$

2 Iterativno računanje rešitve igre

V projektu sem implementirala več iterativnih algoritmov, ki nam vrnejo optimalno in vrednost igre ter analizirala kako hitra je konvergenca posameznih metod. Nato bom naredila tudi spletno aplikacijo, s pomočjo katere dobimo vrednost matrične igre in ki nam vrne analizo konvergence različni iterativnih metod pri reševanju izbrane matrične igre. Za implementacijo sem uporabila programski jezik *Python*, pri analizi metod pa sem si pomagala tudi s programskim jezikom *R*. Celotna koda, ki vsebuje vse implementacije iterativnih algoritmov, se nahaja v datoteki *projekt.py*.

2.1 Metoda I

Naj bo $V(t)$ vektor in $v_j(t)$ naj bo njegova j-ta komponenta. Označimo $\max V(t) = \max_j v_j(t)$ in $\min V(t) = \min_j v_j(t)$. Naj bo sistem (U, V) sestavljen iz zaporedja n-dimenzionalnih vektorjev $U(0), U(1), \cdots$ in zaporedja m-dimenzionalnih vektorjev $V(0), V(1), \cdots$ in naj velja $\min U(0) = \max V(0)$ in $U(t+1) = U(t) + A_{i\cdot}$ ter $V(t+1) = V(t) + A_{\cdot j}$, pri čemer i in j zadoščata pogojem:

$$v_i(t) = \max V(t),\ u_j(t) = \min U(t).$$

Potem vrednost igre v dobimo tako, da za njo velja:

$$\lim_{t\to\infty} \frac{\min U(t)}{t} = \lim_{t\to\infty} \frac{\max V(t)}{t} = v$$

Funkcija v kodi, ki reši zgoraj opisani problem se imenuje *vrednostIgre* in sprejme dva argumenta - *matrika* in *steviloIteracij*. Začetni približek sem generirala iz enakomerne porazdelitve $U(a, b)$, kjer sta a in b najmanjši in največji element matrike po absolutni vrednosti. Funkcija vrne vrednost matrične igre, pri čemer velja omeniti, da sem za vrednost igre vzela povprečje od $\frac{\min U(t)}{t}$ in $\frac{\max V(t)}{t}$.

2.2 Metoda II

Najprej uvedimo še nekaj nove notacije. Naj velja:

$$A_i = (a_{1i}, \cdots, a_{mi}, \underbrace{0, \cdots, 0}_{n-\text{komponent}}, -1),$$

$$A_{0i} = (\underbrace{1, \cdots, 1}_{m-\text{komponent}}, \underbrace{0, \cdots, 0}_{n-\text{komponent}}, 0)$$

$$\text{za } i = 1, \cdots, n$$

in

$$A_i = (\underbrace{0, \cdots, 0}_{m-\text{komponent}}, -a_{i-n,1}, \cdots, -a_{i-n,n}, 1),$$

$$A_{0i} = (\underbrace{0, \cdots, 0}_{m-\text{komponent}}, \underbrace{1, \cdots, 1}_{n-\text{komponent}}, 0)$$

$$\text{za } i = n+1, \cdots, m+n.$$

Definirajmo še vektor, ki predstavlja rešitev igre: $Z^* = (X^*, Y^*, v)$. Z^* mora poleg 1, 2, 3, 4, ustrezati še pogoju:

$$A_i \cdot Z^* \geq 0 \text{ za } i = 1, \cdots, m+n. \tag{5}$$

Metoda se začne s poljubnim vektorjem $Z^{(1)}$, ki zadošča 1, 2, 3 in 4. Sedaj predpostavimo, da smo prišli na k-ti korak iteracije, in dobili vektor $Z^{(k)}$, ki ustreza 1, 2, 3 in 4. Če velja tudi 5, je $Z^{(k)}$ rešitev igre in smo zaključili. Sicer pa naj bo j_k tak indeks, da bo veljalo $A_{j_k} \cdot Z^{(k)} \leq A_i \cdot Z^{(k)}$ za vse $i = 1, \cdots, m+n$. Če obstaja več takih indeksov, lahko poljubno izberemo. Če torej poznamo indeks j_k, lahko dobimo nov vektor $\bar{Z}^{(k+1)} = (\bar{X}^{(k+1)}, \bar{Y}^{(k+1)}, \bar{v}^{(k+1)})$ na sledeči način:

$$\bar{Z}^{(k+1)} = Z^{(k)} + \alpha B_{j_k} + \beta B_{0j_k},$$

kjer je

$$\alpha = -Z^{(k)} \cdot B_{j_k}[1 - \cos^2 \theta_{j_k}]^{-1},$$
$$\beta = b_{0j_k} - [Z^{(k)} + \alpha B_{j_k}] \cdot B_{0j_k},$$
$$b_{0j_k} = \frac{1}{(A_{0j_k} \cdot A_{0j_k})^{1/2}},$$
$$B_{j_k} = \frac{A_{j_k}}{(A_{j_k} \cdot A_{j_k})^{1/2}},$$
$$B_{0j_k} = \frac{A_{0j_k}}{(A_{0j_k} \cdot A_{0j_k})^{1/2}}$$

in

$$\cos \theta_{j_k} = \frac{A_{0j_k} \cdot A_{j_k}}{(A_{0j_k} \cdot A_{0j_k})^{1/2}(A_{j_k} \cdot A_{j_k})^{1/2}}.$$

Sedaj predpostavimo, da velja $j_k < (n+1)$. (V primeru, da bi bil $j_k \geq n+1$, bi komponente x ostale nespremenjene, postopek, opisan v nadaljevanju, pa bi veljal za y komponente.) Če $\bar{Z}^{k+1}$ ustreza 5, potem nastavimo $Z^{k+1} = \bar{Z}^{k+1}$, v nasprotnem primeru pa moramo, da dobimo Z^{k+1}, narediti še nekaj korakov. Najprej vse negativne x-komponente vektorja $\bar{Z}^{k+1}$ nastavimo na 0. Predpostavimo, da so $\bar{x}_1^{(k+1)}, \cdots, \bar{x}_r^{(k+1)}$, $r < m$ negativne komponente vektorja $\bar{Z}^{(k+1)}$. Nato izračunamo vse vsote $\bar{x}_i^{(k+1)} + \frac{\sum_{i=1}^{r} \bar{x}_i^{(k+1)}}{m-r}$ za $i = r+1, \cdots, m$. Za vsak tak i, za katerega je vsota negativna, nastavimo $x_i^{(k+1)} = 0$. Če nobena vsota ni negativna, lahko tvorimo preostanek vektorja $Z^{(k+1)}$. Spet predpostavimo, da so nekatere vsote za $i = r+1, \cdots, r+s$ negativne. Ponovno izračunamo vsote $\bar{x}_i^{(k+1)} + \frac{\sum_{i=1}^{r+s} \bar{x}_i^{(k+1)}}{m-(r+s)}$ za $i = r+s, \cdots, m$. Če nobena vsota ni negativna, tvorimo preostanek vektorja $Z^{(k+1)}$, sicer pa ponavljamo zgornji postopek, dokler nobena od vsot ni negativna.

Predpostavimo, da za $i = 1, \cdots, t$ velja, da je $\bar{x}_i^{(k+1)} \leq 0$ ali pa, da je $\bar{x}_i^{(k+1)}$ tak, da je zanj katera od zgoraj definiranih vsot negativna. Potem lahko vektor $Z^{(k+1)}$ tvorimo na sledeči način:

$$x_1^{(k+1)} = \cdots = x_t^{(k+1)} = 0,$$

$$x_i^{(k+1)} = \bar{x}^{(k+1)} + \frac{\sum_{i=1}^{t} \bar{x}_i^{(k+1)}}{m-t} \text{ za } i = t+1, \cdots, m,$$
$$y_j^{(k+1)} = \bar{y}_j^{(k+1)} \text{ za } j = 1, \cdots, n,$$
$$v^{(k+1)} = \bar{v}^{(k+1)}.$$

Opisana metoda je implementirana s funkcijo *iteracija2* in sprejme dva argumenta - *steviloIteracij* in *matrika*. Za začetne približke sem vzela $x_0 = (1/m, \cdots, 1/m)$, $y_0 = (1/n, \cdots, 1/n)$, v_0 pa sem generirala iz enakomerne porazdelitve $U(a, b)$, kjer sta a in b najmanjši in največji element matrike po absolutni vrednosti. Funckija *iteracija2* po k-iteracijah vrne vektor $Z^k = (X^k, Y^k, v^k)$, ki predstavlja rešitev igre.

2.3 Metoda 3

Naslednja metoda, ki sem jo implemenitrala se imenuje statistična Brownova metoda. Temelji na ideji, da so sedanje odločitve odvisne od zgodovine. 1. igralec najprej igra poljubno čisto strategijo X_{i_1}, pri čemer je i_1 komponenta iz množice $(1, \cdots, m)$ enaka 1, ostale komponente pa so enake 0. 2. igralec nato začne z vektorjem akumulativnih vsot $A^{(1)} = [a_1^{(1)}, \cdots, a_n^{(1)}]$, kjer $A^{(1)}$ predstavlja i_1-to vrstico matrike. 2. igralec nato igra čisto strategijo Y_{j_1}, pri čemer je j_1 tak indeks iz množice $(1, \cdots, n)$, da velja, da je $a_{j_1}^{(1)}$ najmanjša komponenta vektorja $A^{(1)}$. Če je minimumov več načeloma vzamemo poljubnega, pri implementaciji pa sem vzela prvi minimum. 1. igralec nato začne z vektorjem akumulativnih vsot $B^{(1)} = [b_1^{(1)}, \cdots, b_m^{(1)}]$, pri čemer je $B^{(1)}$ j-ti stolpec matrike. 1. igralec tako uporabi čisto strategijo X_{i_2}, pri čemer je i_2 tak indeks iz množice $(1, \cdots, m)$, da velja, da je $b_{j_2}^{(1)}$ največja komponenta vektorja $B^{(1)}$ Če je maksimumov več načeloma vzamemo poljubnega, pri implementaciji pa sem vzela prvi maksimum. 1. igralec nato vektorju $A^{(1)}$ prišteje i_2 - to vrstico matrike, zatem pa 2. igralec igra na enak način kot prej. Na k-tem koraku imamo vektorja akumulativnih vsot $A^{(k)} = [a_1^{(k)}, \cdots, a_n^{(k)}]$ in $B^{(k)} = [b_1^{(k)}, \cdots, b_m^{(k)}]$. 1. igralec nato igra čisto strategijo X_{i_k+1}, pri čemer je i_{k+1} tak indeks, da je $b_{j_k+1}^k$ maksimalna komponenta vektorja $B^{(k)}$. $A^{(k+1)}$ dobimo tako, da vektorju $A^{(k)}$ prištejemo $j_{(k+1)}$-to vrstico matrike izplačil. 2. igralec tako igra čisto strategijo $Y_{j(k+1)}$, pri čemer je $j_{(k+1)}$ tak indeks, da je $a_{j(k+1)}^{(k+1)}$ najmanjša komponenta vektorja $A^{(k+1)}$. Postopek ponavljamo. Dokazati se da, da zaporedje $\frac{\sum_{n=1}^{k} X_{i_n}}{k}$ ali konvergira k optimalni strategiji X^* ali pa ima tako podzaporedje. Prav tako tudi zaporedje $\frac{\sum_{n=1}^{k} Y_{i_n}}{k}$ ali konvergira k optimalni strategiji Y^* ali pa ima podzaporedje, ki konvergira.

	1. metoda		2. metoda				3. metoda			
	čas	v	čas	v	x	y	čas	v	x	y
k = 1	0.000166	1.9	0.003655	1.846153						
k = 2	0.000178	0.95	0.001894	0.739316						
k = 10	0.000455	0.2847420	0.003198	0.536365						
k = 50	0.001109	0.033	0.015611	0.027247						
k = 100	0.002001	0.023	0.037982	0.005604						
k = 500	0.011203	0.0031	0.182010	0.000926						
k = 1000	0.012737	0.0023	0.304981	0.000210						

Vrednost igre pri tem algoritmu dobimo kot $\max_{k=1,2,\cdots} \frac{\min_{i\in(1,\cdots,n)} a_i^{(k)}}{k}$ in kot $\min_{k=1,2,\cdots} \frac{\max_{i\in(1,\cdots,m)} b_i^{(k)}}{k}$.

Opisana metoda je implementirana s funkcijo *iteracijaBrown* in sprejme 2 argumenta - *matrika* in *steviloIteracij*. Funkcija vrne optimalni strategiji po k-tih korakih iteracije $X^{(k)}$ in $Y^{(k)}$, ter vrednost igre $v^{(k)}$. Z vrednost igre sem vzela povprečje $\max_{k=1,2,\cdots} \frac{\min_{i\in(1,\cdots,n)} a_i^{(k)}}{k}$ in $\min_{k=1,2,\cdots} \frac{\max_{i\in(1,\cdots,m)} b_i^{(k)}}{k}$

3 Analiza konvergence, časovna zahtevnost in rezultati

Naj najprej povem, da sem vsak poskus ponovila 10-krat in nato vzela povprečje rezultatov. Najprej si bomo pogledali, kako na treh matrikah različnih dimenzij delujejo zgoraj opisani algoritmi. Začnimo z matriko dimenzije 3×2:

$$\begin{bmatrix} 4 & 3 \\ 2 & 4 \\ 5 & 2 \end{bmatrix}$$

Vrednost prve matrične igre je $v = \frac{10}{3}$, medtem, ko je mešano Nahevo ravnovesje $X = [\frac{2}{3}, \frac{1}{3}]$ in $Y = [\frac{1}{3}, \frac{2}{3}]$.

Literatura

[1] J. Robinson, *An Iterative Method of Solving a Game*, Annals of Mathematics, **1951** strani od 296 do 301. Dostopno na: https://www.jstor.org/stable/1969530

[2] R. J. Jirka, *An iterative method for finding a solution to a zero-sum two person rectangular game*, **1959**.