

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 naj označuje i -to vrstico, A_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 verjetnostjo y_j , pri čemer velja:

$$x_i \geq 0, \quad (1)$$

$$\sum x_i = 1, \quad (2)$$

$$y_i \geq 0, \quad (3)$$

$$\sum y_i = 1. \quad (4)$$

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 $X = (x_1, \dots, x_m)$ in $Y = (y_1, \dots, y_n)$ 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 $U(0), U(1), \dots$ in zaporedja m -dimenzionalnih vektorjev $V(0), V(1), \dots$ in naj velja $\min U(0) = \max V(0)$ in $U(t+1) = U(t) + A_i$ ter $V(t+1) = V(t) + A_j$, pri čemer i in j zadoščata pogojem:

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

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

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

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:

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

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

za $i = 1, \dots, n$

in

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

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

za $i = n+1, \dots, m+n$.

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

$$A_i \cdot Z^* \geq 0 \text{ za } i = 1, \dots, m+n. \quad (5)$$

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 $i = 1, \dots, m+n$. 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 $\bar{x}_1^{(k+1)}, \dots, \bar{x}_r^{(k+1)}$, $r < m$ 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 $i = r+1, \dots, m$. 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 $i = r+1, \dots, r+s$ negativne. Ponovno izračunamo vsote $\bar{x}_i^{(k+1)} + \frac{\sum_{i=1}^{r+s} \bar{x}_i^{(k+1)}}{m-(r+s)}$ za $i = r+s, \dots, m$. 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 $i = 1, \dots, t$ 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:

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

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

Opisana metoda je implementirana s funkcijo *iteracija2* in sprejme dva argumenta - *steviloIteracij* in *matrika*. Za začetne približke sem vzela $x_0 = (1/m, \dots, 1/m)$, $y_0 = (1/n, \dots, 1/n)$, v_0 pa sem generirala iz enakomerne porazdelitve $U(a, b)$, kjer sta a in b najmanjši in največji element matrike po absolutni vrednosti. Funkcija *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 $(1, \dots, m)$ enaka 1, ostale komponente pa so enake 0. 2. igralec nato začne z vektorjem akumulativnih vsot $A^{(1)} = [a_1^{(1)}, \dots, a_n^{(1)}]$, 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 $(1, \dots, n)$, 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 $B^{(1)} = [b_1^{(1)}, \dots, b_m^{(1)}]$, 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 $(1, \dots, m)$, 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 $A^{(k)} = [a_1^{(k)}, \dots, a_n^{(k)}]$ in $B^{(k)} = [b_1^{(k)}, \dots, b_m^{(k)}]$. 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.

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

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 $\max_{k=1,2,\dots} \frac{\min_{i \in (1,\dots,n)} a_i^{(k)}}{k}$ in $\min_{k=1,2,\dots} \frac{\max_{i \in (1,\dots,m)} b_i^{(k)}}{k}$.

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 $X = [\frac{2}{3}, \frac{1}{3}, 0]$ in $Y = [\frac{1}{3}, \frac{2}{3}]$.

Nato si oglejmo matriko velikosti 7×7 :

$$\begin{bmatrix} 0 & -1 & -2 & 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 4 & 0 & 0 & -1 \\ 2 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & -4 & 1 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 0 & 0 & -3 & 1 \\ -2 & 0 & 0 & 1 & 3 & 0 & -1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}$$

Ker je 2. matrična igra simetrična, je njena vrednost $v = 0$. Optimalni strategiji sta $X = [0.1111, 0.1111, 0.1667, 0.0556, 0.1667, 0.0556, 0.3333]$ in $Y = [0.1111, 0.1111, 0.1667, 0.0556, 0.1667, 0.0556, 0.3333]$. Naslednja tabela prikazuje rezultate, dobljene pri vseh treh opisanih metodah.

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>

	1. metoda		2. metoda				3. metoda			
	čas	v	čas	v	x	y	čas	v	x	y
k = 1	0.0006	6.95	0.0020	3	0.33333 0.33333 0.33333	0 1	0.0009	1.7500	0.5 0.5 0	0 0.5
k = 2	0.0006	4.375	0.0020	3.49020	0.33333 0.33333 0.33333	0.5 0.5	0.0014	1.6600	0.66667 0.33333 0	0.33333 0.33333
k = 10	0.0007	3.55	0.0155	3.31150	0.34199 0.45804 0.19997	0.34425 0.65575	0.0020	2.1001	0.45454 0.45454 0	0.27273 0.63636
k = 50	0.0016	3.451	0.0156	3.33100	0.47646 0.40436 0.11918	0.33100 0.66900	0.0030	3.2310	0.64706 0.33333 0.01961	0.35294 0.62746
k = 100	0.0031	3.3661	0.0369	3.33147	0.51447 0.39017 0.09536	0.33146 0.66854	0.0049	3.3109	0.66337 0.32673 0.00990	0.33663 0.65347
k = 500	0.0107	3.3281	0.0953	3.33301	0.64108 0.34289 0.01603	0.33302 0.66670	0.0156	3.32335	0.66467 0.33333 0.00200	0.33134 0.66667
k = 1000	0.0173	3.3339	0.1207	3.33330	0.66391 0.33436 0.00173	0.33330 0.66670	0.0312	3.32867	0.66633 0.33257 0.00010	0.33367 0.66533
k = 15000	0.2612	3.3359	2.4360	3.33333	0.66667 0.33333 0	0.33333 0.66667	0.4397	3.33304	0.66656 0.33338 0.00007	0.33338 0.66656

Tabela 1: Primerjava rezultatov za 1. igro

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

	1. metoda		2. metoda				3. metoda			
	čas	v	čas	v	x	y	čas	v	x	y
k = 1	0.00017	1.9	0.00365	1.84615	0.01282 0.16667 0.16667 0 0.16667 0.16667 0.32051	0.14286 0.14286 0.14286 0.14286 0.14286 0.14286 0.14286	0.00165	0.25	0 0 0.5 0 0 0 0.5	0.5 0 0 0 0 0 0
k = 2	0.00018	0.95	0.00489	0.73931	0.21520 0 0 0.20238 0.36904 0 0.21337	0.14286 0.14286 0.14286 0.14286 0.14286 0.14286 0.14286	0.00270	0.16667	0 0.33333 0.33333 0.33333 0 0 0 0.33333	0.33333 0 0 0.33333 0 0 0 0
k = 10	0.00045	0.28474	0.00319	0.53636	0.06687 0.22339 0.24321 0 0.21376 0 0.25274	0.14286 0.14286 0.14286 0.14286 0.14286 0.14286 0.14286	0.00575	0.04545	0.18182 0.27273 0.09091 0.09091 0.27273 0 0.09091	0.09091 0.27273 0.09091 0.09091 0.09091 0 0.36364
k = 50	0.00111	0.033	0.01561	0.02724	0.11101 0.10108 0.16020 0.06888 0.16563 0.04690 0.34627	0.10855 0.14050 0.14894 0.05297 0.16978 0.11520 0.26410	0.00418	0.00980	0.11765 0.09804 0.27451 0.07843 0.07843 0.03922 0.31373	0.11765 0.13725 0.17647 0.05882 0.07843 0.07843 0.33333
k = 100	0.00200	0.023	0.03798	0.00560	0.11286 0.11280 0.16732 0.05262 0.16899 0.05386 0.33153	0.11161 0.13054 0.14982 0.05716 0.16783 0.07269 0.31035	0.00557	0.00495	0.10891 0.09901 0.15842 0.03960 0.17822 0.08911 0.32673	0.12871 0.11881 0.18812 0.04951 0.15842 0.03960 0.30693
k = 500	0.01120	0.0031	0.18201	0.00092	0.111015 0.111365 0.166570 0.055421 0.166630 0.055414 0.333585	0.11136 0.11839 0.16141 0.05547 0.16639 0.05588 0.33111	0.03899	0.00099	0.09980 0.14371 0.16567 0.05589 0.16567 0.05190 0.31737	0.12575 0.15369 0.13573 0.04591 0.14970 0.06387 0.32335
k = 1000	0.01273	0.0023	0.30498	0.00021	0.111174 0.111040 0.166691 0.055660 0.166655 0.055518 0.333259	0.11109 0.11311 0.16518 0.05562 0.16665 0.05559 0.33277	0.04474	0.00050	0.11289 0.11089 0.17183 0.04695 0.16484 0.05594 0.33666	0.11588 0.11189 0.17083 0.04595 0.17383 0.04895 0.33167
k = 15000	0.21999	0.00011	4.83584	3.1e-17	0.11111 0.11111 0.16667 0.05556 0.16667 0.05556 0.33333	0.11111 0.11111 0.16667 0.05556 0.16667 0.05556 0.33333	0.58264	3.3e-05	0.11312 0.11112 0.16732 0.05466 0.16299 0.05520 0.33558	0.10752 0.11179 0.16739 0.05786 0.16545 0.05446 0.33544

Tabela 2: Primerjava rezultatov za 2. igro