МАТЕМАТИЧКИ ОМНИБУС, 9 (2021), 9 – 21
http://im-pmf.weebly.com/matematicki-omnibus-kniga-9.html

КОРИСТЕЊЕ НА МАТРИЧНА ФАКТОРИЗАЦИЈА ВО СИСТЕМИ ЗА ПРЕПОРАЧУВАЊЕ

Јаков Митровски [1]

Марија Михова [1]

Под систем за препорачување се подразбира систем кој тежи да ја предвиди оцената или преферирањето на одреден објект од страна на специфичен корисник. Последните години ваков тип системи се неизбежен дел од платформите кои се трудат да го погодат вкусот на своите корисници и да им ги понудат оние производи или содржини кои ним најмногу би им се допаднале. Затоа се користат во многу области, од понуда на музика, интернет содржини, курсеви за слушање, интернет продавници и слично. Најпознат пример за ваков систем е платформата Нетфликс (Netflix) која врши селекција на филмови и серии кои би им се допаднале на нивните корисници. Имено, ваквите системи добија најголема популарност кога оваа компанија го објави познатиот натпревар “The Netflix Prize”, [1], во кој понудија еден милион долари на оној кој ќе го подобри постоечкиот систем за препораки за 10%. Во овој труд ќе биде анализирано решението кое победи на натпреварот, кое се базира на факторизација на матрици, односно запишување на матрицата од веќе дадени оцени, како производ од две други матрици.

Еден систем за препорака на филмови има задача да селектира листа на филмови кои би му се допаднале на конкретниот корисник, врз база на неговите преференции. Неговите преференции се оценуваат врз основа на оцените кои тој веќе ги има дадено за одредени филмови или пак неговата историја на гледани филмови. Селекцијата на филмовите предвид ги зема сличноста помеѓу филмовите и сличноста помеѓу корисниците, и смета дека слични корисници би дале слични оцени за слични филмови. Различните приоди се разликуваат во начинот на оценување на овие сличности, но сите тие, ова оценување го прават преку анализирање на историските податоци од сите корисници и филмови, кои претходно ги имаат на располагање. Па, системот има за цел точно да предвиди каква оцена би дал еден корисник за филм кој тој го нема гледано, и да му ги препорача филмовите за кои тој би дал висока оцена.

1. ВИДОВИ СИСТЕМИ ЗА ПРЕПОРАЧУВАЊЕ

Во последнава деценија, системите за препорачување наоѓаат примена во широк спектар од области кои му се од интерес на човекот. Генерално овие системи се користат за да се предвиди оцената која корисникот би ја дал за множество на ставки преку анализа на неговото претходно оценување и на корисници кои имаат слични карактеристики на корисникот кој е разгледуван. Целта на еден систем за препорачување е да успее да ја предвиди оцената која корисникот би ја дал за одредена ставка, за која важи дека тој не се сретнал претходно со таа ставка, односно, не ја оценил. Постојат два метода кои се користат при изработка на еден ваков систем, и тоа:

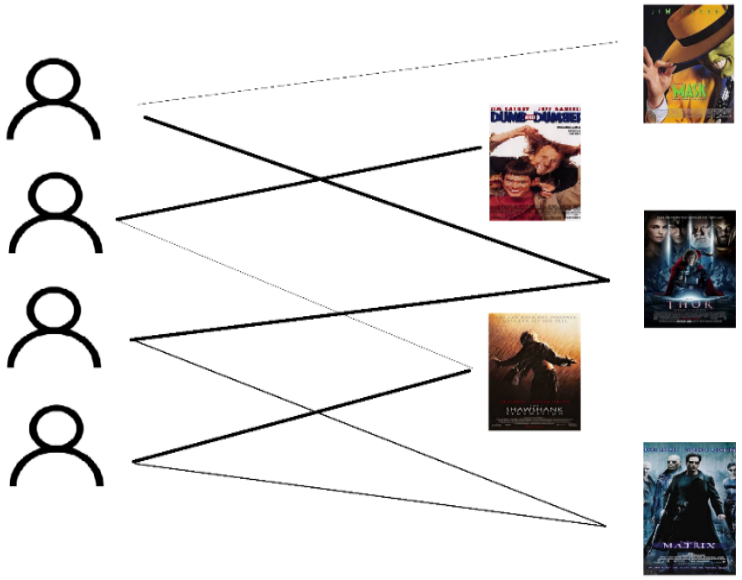
- Методи за колаборативно филтрирање.
- Методи базирани на содржина, [6].

Методите за колаборативно филтрирање како претпоставка земаат дека слични корисници имаат слични преференции. Па така, врз основа на тоа каква оцена поставиле корисниците слични на даден корисник *u*, за ставката *v*, може да се предвиди оцената која корисникот *u* би ја дал на таа ставка *v*. Со цел ова да функционира, најголемиот дел од ваквите методи користат мерка која ја дефинира сличноста помеѓу корисниците [5]. Од друга страна, методите базирани на содржина работат поинаку. Кај овие методи предвидувањето на оцената која ја дал корисникот *u* за ставката *v* се темели на претходните оцени што корисникот *u* ги дал на други ставки кои се слични на ставката *v*. Методот кој е искористен од страна на платформата Нетфликс (Netflix) спаѓа во групата на методи за колаборативно филтрирање, дополнително користејќи факторизација на матрици за пресметување на предвидените вредности на оценките.

2. ФОРМУЛАЦИЈА НА ПРОБЛЕМОТ НА ПРЕПОРАЧУВАЊЕ

Проблемот на препорачување на филмови, графички може да се прикаже како што е претставено на Слика 1., каде корисниците се претставени на левата страна, а филмовите на десната. Корисниците се поврзани со филмовите со тенки или со дебели линии. Доколку линијата е тенка тој корисник го оценил филмот со ниска оценка, а доколку лини-

јата е дебела, корисникот го оценил филмот високо. Систем за препорачување има задача да го одреди видот („дебелината“) на оние линии кои не постојат, односно, да се најдат оценките кои дадени корисници би ги дале, за филмови кои тие не ги оцениле.



Слика 1. Графички приказ на проблемот за преферирање на филмови.

За подобро моделирање на проблемот, може да се искористи матрица R чии димензии ќе бидат $n \cdot m$ каде n е бројот на корисници, а m е бројот на филмови.

	Ф1	Ф2	Ф3	Ф4	Ф5
К1	3		1		1
К2	1		4	1	
К3	3	1		3	1
К4		3		4	4

Табела 1. Пример за матрицата R со оценки на четири корисници (К1, К2, К3, К4) за пет филма (Ф1, Ф2, Ф3, Ф4, Ф5).

Секоја од редиците на матрицата R опишува еден корисник, секоја колона – еден филм, а вредноста во ќелијата (i, j) на матрицата е број кој ја претставува оцената на корисникот i за филмот j. Доколку ќелијата (i, j) нема вредност, тоа значи дека корисникот i не го оценил филмот j.

Табела 1 претставува една таква матрица *R*. Може да се забележи дека, на пример, првиот корисник (К1), го оценил петтиот филм (Ф5) со оцена 1, а од друга страна, третиот корисник (К3), не го оценил третиот филм (Ф3).

3. МЕТОД НА МАТРИЧНА ФАКТОРИЗАЦИЈА

Претходно споменатиот метод на матрична факторизација е познат како еден од најефикасните методи кога станува збор за прецизноста на предвидувањето на оценките во контекст на системите за препорачување [2, 4]. Овој метод се труди да ја дополни матрицата *R* со тоа што ќе ги допише оние оцени кои корисниците би ги дале за филмови кои тие не ги оцениле. Дополнително, работи под претпоставка дека матрицата *R* не е празна, и како негова основа се земаат податоците запишани во *R* во облик *(kor, fil, oce),* каде *kor* е корисник, *fil* е филм, а *oce* е оценката која корисникот *kor* ја дал за филмот *fil*. За да може да се применуваат некои матрични трансформации или одредени математички операции, потребно е матрицата *R* да е целосно пополнета со целобројни оцени од 1 до 5. Најчесто местата каде што нема вредност во матрицата *R* се заменуваат со 0.

Методот на матрична факторизација ја оценува матрицата *R* на начин што цели да најде две матрици *U* и *V* чии димензии се помали од димензиите на матрицата *R*. Поточно, матрицата *U* има помалку колони, а матрицата *V* има помалку редици од матрицата *R*. Производот на матриците *U* и *V* треба да даде што е можно поточни вредности за оние податоци кои се веќе познати во матрицата *R*. Со множење на матриците *U* и *V*, ќе се добијат и вредностите кои недостасуваат во матрицата *R*, па овие вредности се земаат како предвидени вредности за точните оцени кои корисниците би ги дале за филмвите кои тие не ги оцениле.

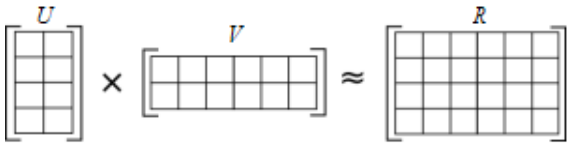
Интуитивно, овој метод претпоставува дека постојат одредени карактеристики кои што ги поседуваат и филмовите и корисниците и дека оценката која даден корисник ја дава за даден филм не е случајна, туку се базира на овие карактеристики уште наречени и латентни карактеристики (*latent features*). На пример, оние корисници кои преферираат комедии, би дале повисока оцена на филмовите од тој жанр, а пониска за филмовите кои се хорор, акција и слично. Според тоа,

латентни карактеристики би биле жанрот на филмот, главниот актер, дали филмот е анимиран и сл. Па така, доколку се опишат корисниците и филмовите со помош на овие латентни карактеристики (кои немаат секогаш одредено значење), тогаш ќе може да се предвиди оцената која корисникот би ја дал за даден филм. Затоа карактеристиките кои соодветствуваат на корисникот треба да се поклопат со оние кои се соодветни за филмот.

Во конкретна примена на методот, бројот на латентни карактеристики не е однапред определен, односно не се знае ниту кои се, а ниту колку вакви карактеристики го опишуваат проблемот. Затоа се нарекуваат латентни, односно скриени. Но, всушност овој метод воопшто не го интересира кое е значењето на карактеристиките, туку само нивниот број. Затоа, во конкретна апликација се експериментира со бројот на латентни карактеристики. Па така, бројот на овие карактеристики се бира така да се добие оптимална брзина на алгоритамот и разумна конвергенција на грешката помеѓу предвидените оцени и оние кои веќе ги знаеме. Тоа би значело дека во општ случај, се пробуваат повеќе различни вредности за бројот на латентни карактеристики и потоа се споредуваат добиените резултати.

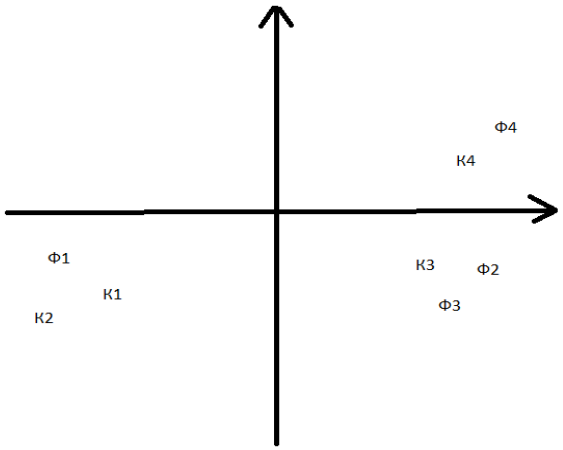
Нека n е бројот на корисници и m е бројот на филмови. Тогаш, матрицата R има димензии $n \cdot m$ и ги содржи сите оцени кои биле дадени од страна на корисниците за одредени филмови, или вредност 0, ако корисникот не дал оцена за соодветниот филм. Да претпоставиме дека се одбрани k латентни карактеристики. На Слика 2. е прикажан методот на факторизација на матрици, чија цел е да се најдат две матрици U (со големина $n \cdot k$) и V (со големина $k \cdot m$) такви што, при нивно множење ќе се добие матрица приближно еднаква на R, односно оценка за R, која ја бележиме со $\hat{R}$. Притоа, матрицата U претставува врска помеѓу корисниците и латентните карактеристики, каде што редицата i ги опишува важностите на латентните карактеристики за i-тиот корисник (Кi). Од друга страна, V е врска помеѓу латентните карактеристики и филмовите, каде колоната j ги опишува важностите на латентните карактеристики за j-иот филм (Фj).

$$R \approx U * V = \hat{R}. \qquad (1)$$



Слика 2. Пресметувањето на матрицата R^е дефинирано со равенството (1).

Главната идеја во методот на матрична факторизација е со помош на матриците U и V да се опишат сите корисници и филмови користејќи ги латентните карактеристики. Односно, секој од корисниците и филмовите да ги запишеме како вектори во k-димензионален простор (чии оски се латентните карактеристики). Во зависност од тоа колку растојанието помеѓу два вектори е помало, толку тие имаат слични латентни карактеристики. Па така, доколку еден корисник kor_i и филм fil_j се на мало растојание во овој k-димензионален простор, тогаш се очекува корисникот kor_i да го оцени фимлот fil_j високо.



Слика 3. Корисници и филмови мапирани со помош на 2 латентни карактеристики

Нека една од вредностите кои недостасуваат во матрицата R е на позиција (i, j). За да се оцени оваа вредност, односно за да се предвиди вредноста на оцената која корисникот u_i би ја дал за филмот v_j, треба да се пресмета скаларниот производ на соодветните вектори од матриците U и V на следниот начин:

$$\hat{r}_{ij} = u_i v_j = \sum_{p=1}^{k} u_{ip} v_{pj}. \qquad (2)$$

За да се пресмета резултатот од формулата (2) најпрво треба да се направи оценка и за самите матрици *U* и *V*. Еден од начините со кои може да се пристапи на овој проблем е да се иницијализираат двете матрици со случајни вредности, а потоа итеративно да се препресметуваат, така што во секој чекор се повеќе би се приближувале до нивната точна вредност. За таа цел треба да се користи функција на грешка, која дава мерка за тоа колку оценетата матрица $\hat{R}$ отстапува од дадената матрица R. Итеративните промени треба да се прават во насока на минимизирање на функцијата на грешка која е дефинирана со равенството (3).

$$E = \sum_{i=1}^{n}\sum_{j=1}^{m} e_{ij}^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(r_{ij} - \hat{r}_{ij}\right)^2, \qquad (3)$$

за секои i, j такви што $r_{ij} \neq 0$.

Како функција на грешка најчесто се зема сумата на квадратното отстапување на предвидените од вистинските вредности. Заради тоа овде се зема ова отстапување на оценката од вистинската вредност (реалната оцена на корисникот *i* за филмот *j*), за секоја од позициите (*i, j*) на матрицата *R*. При пресметување на вредноста на функцијата на грешка, важно е да се напомене дека оние ќелии за кои нема вредности во матрицата *R* се игнорираат. Ваквиот итеративен процес на ажурирање на вредностите на матриците, следејќи го минимумот на дефинирана функција на грешка, е познат под терминот „gradient descent“ и се користи да се најде локален минимум на дадена функција.

За да се минимизира функцијата на грешка, треба да се знае во која насока и колку да се променат вредностите u_{ip} и v_{pj} (да се зголемат или да се намалат). Затоа, со пресметување на парцијалните изводи на e_{ij}^2 во однос на промелнивите u_{ip} и v_{pj} (со помош на равенките (4.1) и (4.2)) може да се заклучи колку функцијата на грешка е далеку од минимумот.

$$\frac{d}{du_{ip}} e_{ij}^2 = -2\left(r_{ij} - \widehat{r_{ij}}\right)\left(v_{pj}\right) = -2e_{ij}v_{pj} \qquad (4.1),$$

$$\frac{d}{dv_{pj}} e_{ij}^2 = -2\left(r_{ij} - \widehat{r_{ij}}\right)\left(u_{ip}\right) = -2e_{ij}u_{ip} \ . \qquad (4.2)$$

Откако ќе се пресметаат овие вредности, може да се формулираат правилата за менување на вредностите на u_{ip} и v_{pj}:

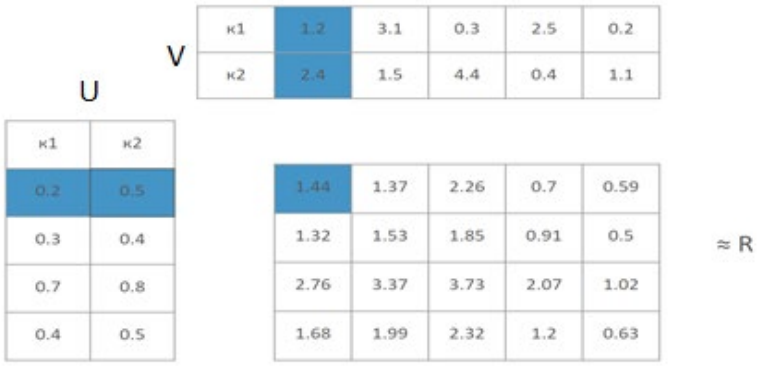
$$u'_{ip} = u_{ip} - \alpha \frac{d}{du_{ip}} e_{ij}^2 = u_{ip} + 2\alpha e_{ij} v_{pj}, \quad (4.3)$$

$$v'_{pj} = v_{pj} - \alpha \frac{d}{dv_{pj}} e_{ij}^2 = v_{pj} + 2\alpha e_{ij} u_{ip}. \quad (4.4)$$

Притоа, α претставува константа која ја претставува ратата на учење, односно, стапката со која се приближуваат вредностите u_{ip} и v_{pj} до вредностите за кои функцијата на грешка има минимална вредност. За α вообичаено се одбира ниска вредност, на пример 0.001. Причината за ваквата вредност на α, се должи на идејата да се избегне случај каде што се прескокнува преку минимумот на функцијата и да се заврши осцилирачки околу него. Со други зборови, се одбира ниска вредност за α за да не се направи преголема промена на променливите u_{ip} и v_{pj}, коешто може да резултира со дивергенција на итеративната низа на оценки. Ваквото ажурирање се повторува предодреден број на итерации (на пример 10000) или до момент кога функцијата на грешка достигнала некоја задоволителна вредност (доколку сумата на e_{ij}^2 за секои i,j такви што $r_{ij} \neq 0$ е помала од некоја преодредена вредност, на пример 0,00005). Откако ќе се се направи оценка на матриците *U* и *V*, вредностите кои недостасуваа во матрицата *R*, се добиваат со множење на оценетите матрици $\hat{U}$ и $\hat{V}$.

4. ИЛУСТРАЦИЈА НА МЕТОДОТ

Во овој дел ќе ја илустрираме работата на методот преку примерот на оцени на филмови кој претходно беше даден во Табела 1, којашто ја претставува матрицата *R*. Претпоставуваме дека корисниците и филмовите имаат две латентни карактеристики, па матрицата *R* сакаме да ја претставиме како производ на матрица U со димензија 4 × 2 и матрица *V* со димензија 2×5. На Слика 4. се претставени почетните матрици *U* и *V,* кои се иницијализирани со случајни вредности. Со нивно множење се добива матрицата $\hat{R}$.



Слика 4. Матриците *U* и *V* иницијализирани со случајни вредности и матрицата $\hat{R}$ **.**

Може да се забележи дека предвидената вредност за оценатa на првиот корисник за првиот филм изнесува $\widehat{r_{11}}$=1,44. Оваа вредност отстапува од вистинската која е 3, па затоа треба да се направи ажурирање на вредностите за латентните карактеристики *k*1 и *k*2 на првиот корисник (u_{11}, u_{12}), како и за вредностите за латентните карактеристики *k*1 и *k*2 на првиот филм (v_{11}, v_{12}), По пресметување на парцијалните изводи по променливите u_{11}, u_{12}, v_{11} и v_{12} користејќи ги формулите (4.1) и (4.2) и ажурирање на истите со примена на формулите (4.3) и (4.4) користејќи α = 0,01 (α во овој пример е земено 0,01 само за да бидат значително приметливи промените по една итерација односно за полесна илустрација на едно ажурирање.

При програмското решавање на проблемот, беше користено α = 0,001), новите вредности за променливите ќе изнесуваат 0,24, 0,57, 1,2, 2,5 соодветно. Сега, новата предвидена вредност за r_{11} изнесува $\widehat{r_{11}}$ =1,77, и може да се воочи дека отстапувањето од вистинската вредност е намалено. Следниот чекор е ваквата постапка да се повтори одреден број пати (при решавање на проблемот максмалниот број на итерации кој беше одбран изнесуваше 10000) за секоја од ќелиите во матриците *U* и *V*, со што би се постигнала посакуваната конвергенција, односно, ќе се определи минимумот на функцијата на грешка.

V

к1	3.00	1.00	1.00	3.00	1.00
к2	1.00	2.00	4.00	1.00	3.00

U

к1	к2
1.00	0.00
0.00	1.00
1.00	0.00
1.00	1.00

3.00	1.00	1.00	3.00	1.00
1.00	2.00	4.00	1.00	3.00
3.00	1.00	1.00	3.00	1.00
4.00	3.00	5.00	4.00	4.00

$\hat{R}$

Слика 5. U, V и $\hat{R}$ по 8143 итерации.

По 8143 итерации на препресметување на матрицата $\hat{R}$, ги добиваме оценетите матрици U и V дадени на Слика 5. Бидејќи матрицата од примерот е со помали димензи, функцијата на грешка конвергираше до 0, па нема потреба од понатамошни итерации. На Слика 6 е прикажана споредба помеѓу матриците R и $\hat{R}$, каде со сино се издвоени вредностите кои недостасуваа во матрицата R. Во општ случај, матрицата R е поголема и функцијата на грешка нема да конвергира до 0, тогаш, со процесот на ажурирање на матриците U и V може да се запре откако ќе поминат претходно дефиниран максимален број на итерации. На Слика 7 е даден псевдо кодот од алгоритмот кој беше искористен при решавањето на проблемот.

3		1		1
1		4	1	
3	1		3	1
	3		4	4

R

3.00	1.00	1.00	3.00	1.00
1.00	2.00	4.00	1.00	3.00
3.00	1.00	1.00	3.00	1.00
4.00	3.00	5.00	4.00	4.00

R^

Слика 6. Споредба помеѓу матриците R и $\hat{R}$.

Data: Matricata R, brojot na latentni karakteristiki K, rata na ucenje learningRate, maksimalniot broj na iteracii maxiter
Result: Matricite U i V
iter = 0;
converged = false;
Initialize U, V to UniformReal(0, 5);
while *iter* < *maxiter and not converged* **do**
  errorSquared = 0;
  for *i in* (0,*R.rows*) **do**
    for *j in* (0,*R.columns*) **do**
      $\text{error}_{ij} = (\sum_{p=1}^{K} U_{ip} * V_{pj} - R_{ij})$; $errorSquared += error_{ij}^2$;
      p = 0;
      while $p < K$ **do**
        $U_{ip} = U_{ip} + 2 * learningRate * error_{ij} * V_{pj}$;
        $V_{pj} = V_{pj} + 2 * learningRate * error_{ij} * U_{ip}$;
        p += 1;
      end
    end
  end
  if *errorSquared* < 0.00001 **then**
    convered = true;
  end
  iter += 1;
end
return *U*, *V*

Algorithm 1: *MatricnaFaktorizacija*

Слика 7. Псевдо код од алгоритмот.

5. ПРОСТОРНА И ВРЕМЕНСКА СЛОЖЕНОСТ

Просторната сложеност на овој алгоритам е значително помала од чување на матрицата R во меморија и е O($n \cdot k + m \cdot k$), каде што n е бројот на корисници, m е бројот на филмови, а k е бројот на латентни карактеристики. Ова се должи на фактот дека не мора да се чува матрицата $\hat{R}$, туку таа може да се изведе од матриците U и V со наоѓање на нивниот производ. Оттука, потребно е да се чуваат само матриците U и V. Од друга страна, временската сложеност на алгоритмот е O($n \cdot k^2 \cdot m \cdot t$), каде што n, k и m се претходно дефинираните вредности, а t е бројот на итерации потребни за конвергенција на матриците U и V, но најмногу максималниот број на дозволени итерации. Оваа сложеност директно следува од начинот на кој се пресметуваат матриците U и V, бидејќи за секоја од ќелиите (кои се $m \cdot n$), потребно е t пати, да се направи предвидување чија комплексност е O(k), каде k е бројот на латентни карактеристики.

6. ЗАКЛУЧОК

Алгоритмот кој користи матрична факторизација има широка примена во системите за препорачување и тоа не само во оние кои се користат за препорачување на филмови или серии, туку и во системи кои препорачуваат музика, онлајн продавници кои препорачуваат свои производи [7], предвидени оцени за студенти пред да направат избор за одредени предмети [3] и слично. Овој метод дава многу добри резултати за оценување на непознатите вредности во така наречените ретки матрици, матрици во кои многу мал број на ќелии се познати, па најчесто се користи во такви ситуации. Во овој труд на кратко беше објаснета интуицијата позади основната верзија на алгоритмот за препорака на филмови на платформата Нетфликс и математичкиот апарат за негова релизација, а начинот на работа беше илустриран преку еден едноставен пример.

ЛИТЕРАТУРА

[1] D. Jackson, The Netflix Prize: *How a $1 Million Contest Changed Binge-Watching Forever*, https://www.thrillist.com/entertainment/nation/the-netflix-prize

[2] P. B. Kantor, L. Rokach, F. Ricci, and B. Shapira, *Recommender Systems Handbook*, Springer, 2011.

[3] Lj. Rechkoski, V. Ajanovski and M. Mihova, *Evaluation of grade prediction using model-based collaborative filtering methods*. 2018 IEEE Global Engineering Education Conference, 2018.

[4] P. Symeonidis and A. Zioupos, *Matrix and Tensor Factorization Techniques for Recommender Systems*. Springer, 2016.

[5] G. Tak´acs, I. Pil´aszy, B. N´emeth, and D. Tikk. *Matrix factorization and neighbor based algorithms for the netflix prize problem*. In Proceedings of the 2008 ACM conference on Recommender systems, pages 267–274. ACM, 2008.

[6] H.Tan, J.Guo,Y.Li. 2008. *E-learning Recommendation System*, International Conference on Computer Science and Software Engineering, CSSE, vol. 5, pp 430–433.

[7] M. Zhou, Z. Ding, J. Tang and D.Yin, *Micro Behaviors: A New Perspective in E-commerce Recommender Systems.* In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018.

[1] Факултет за информатички науки и компјутерско инженерство, Универзитет „Св. Кирил и Методиј“, Скопје
ул. „Руѓер Бошковиќ“ 16, 1000 Скопје, Р. Северна Македонија
e-mail: jakovvmitrovski@yahoo.com
marija.mihova@finki.ukim.edu.mk

Примен: 30.3.2021
Поправен: 20.5.2021
Одобрен: 30.5.2021
Објавен на интернет: 29.6.2021