Nenegativna matrična faktorizacija

Marin Jezidžić

29. prosinca 2022.

Sadržaj

1 Uvod

Nenegativna matrična faktorizacija jedna je od istaknutijih metoda u Numeričkoj matematici, te ju u kontekstu računalne znanosti ubrajamo u metodu nenadziranog učenja. Spada pod metode Linearne redukcije dimenzionalnosti, inače skup metoda kojima se linearnim transformacijama podataka postiže kompresija, vizualizacija, odabir bitnih značajki te filtriranje šuma u podatcima.
U primjerima primjene naravno pronalazimo intenzitet piksela slike, frekvenciju riječi u dokumentu(text mining), astrofizički signali, itd.U takvim nam situacijama metode poput SVD dekompozicije neće biti od pretjerane koristi, te tu na scenu stupa NMF. Od njegove pojave 1994. NMF se postupno popularizirao, te se i dan danas izbacuju znanstveni radovi baš na ovu temu, sa nekim revolucionarnim idejama i primjerima primjene u nekim novim područjima.
U ovome radu opisati ćemo metodu, definirati i precizno dokazati relevantne teoretske rezultate bitne za konkretan primjer text mininga(klasteriranja artikala). Tu ćemo dati detaljan uvod i opisati izazove na koje smo naišli, te predložiti rješenja koja ćemo poduprijeti rezultatima koje ćemo usporediti sa usavršenim algoritmom k-means implementiranim u R-u. Kompletan rad programski je odrađen u programskom jeziku R, te će glavni dijelovi biti navedeni u samome radu.

2 Opis metode

Kao što je rečeno u uvodu, slijedi stroga matematička teoretska pozadina kompletne metode.

Definicija 2.1 Za danu nenegativnu matricu $X \in \mathbb{R}^{m\times n}$ i $k \in \mathbb{N}$ takav da je manji od obje dimenzije matrice X, pronađi nenegativne matrice $U \in \mathbb{R}^{m\times k}$ i $V \in \mathbb{R}^{k\times n}$ kako bi se minimizirao izraz

$$J = \frac{1}{2}||X - UV||_F^2$$

Umnožak UV nazivamo nenegativna matrična faktorizacija od X, iako X nije nužno jednak produktu UV, tj. aproksimativno je jednak.

Napomena 2.1.
a) Primijetimo kako u prethodnoj definiciji koristimo Frobeniusovu normu, tj. standardnu matričnu normu: $||X|| = \sqrt{tr(XX^*)}$ gdje X^* označava kompleksno konjuriranu matricu X.
b) Konstanta k se često naziva reducirani rang, no ona će za nas ovdje predstavljati broj klastera.

Sada ćemo iz gornje norme izvesti funkciju greške prema[1].

$$\begin{aligned} J &= \tfrac{1}{2}tr((X - UV^T)(X - UV^T)^T) && (1)\\ &= \tfrac{1}{2}tr(XX^T - 2XVU^T + UV^TVU^T) && (2)\\ &= \tfrac{1}{2}(tr(XX^T) - 2tr(XVU^T) + tr(UV^TVU^T)) && (3)\end{aligned}$$

Ovdje (1) slijedi po definiciji Frobeniusove norme, dok (2) slijedi iz $tr(AB) = tr(BA)$ za neke dvije matrice.

Dakle minimizacija funkcije J se svodi na minimizaciju izraza (3).
Neka je $U = [u_{i,j}], V = [v_{i,j}], U = [U_1, ..., U_k]$.
Sada se gornji problem minimizacije može izreći ovako: Minimiziraj J s obzirom na U i V uz uvjet $u_{i,j} \geq 0, v_{x,y} \geq 0$, gdje su $0 \leq i \leq m$, $0 \leq j \leq k$, $0 \leq x \leq k$, $0 \leq y \leq n$.
Ovako zapisano, ovo je tipičan problem optimizacije. Rješavamo ga koristeći metodu Lagrange-ovih multiplikatora. Neka su $\alpha_{i,j}, \beta_{i,j}$ Lagrangeovi multiplikatori za uvjete $u_{i,j} \geq 0$, $v_{i,j} \geq 0$ respektivno, i $\alpha = [\alpha_{i,j}]$,$\beta =$

$[\beta_{i,j}]$. Lagrangeova funkcija je tada:

$$L = J + tr(\alpha U^T) + tr(\beta V^T)$$

Deriviramo li sada ovaj izraz obzirom na matrice U i V, dobivamo:

$$\frac{\partial L}{\partial U} = -XV + UV^TV + \alpha$$

$$\frac{\partial L}{\partial V} = -X^TU + VU^TU + \beta$$

Koristeći *Kuhn-Tucker* uvjete $\alpha_{i,j}u_{i,j} = 0$ i $\beta_{i,j}v_{i,j} = 0$ dobivamo sljedeće jednadžbe za $u_{i,j}$ i $v_{i,j}$:

$$(XV)_{i,j}u_{i,j} - (UV^TV)_{i,j}u_{i,j} = 0$$
$$(X^TU)_{i,j}v_{i,j} - (VU^TU)_{i,j}v_{i,j} = 0$$

Sređivanjem gornjih jednadžbi dolazimo do takozvanih *updating formulas*:

$$u_{i,j} \leftarrow u_{i,j}\frac{(XV)_{i,j}}{(UV^TV)_{i,j}}$$

$$v_{i,j} \leftarrow v_{i,j}\frac{(X^TU)_{i,j}}{(VU^TU)_{i,j}}$$

Gornje formule predstavljaju takozvani *multiplicative update*. U nastavku rada ćemo priložiti dokaze za njihovu konvergenciju, te da monotono opadaju.

Prvi cilj će nam biti dokazati dvije gornje tvrdnje, te ćemo se onda nakratko pozabaviti algoritamskom složenosti pri implementaciji u programu.

2.1 Monotonost i konvergencija algoritma Hadamardovog produkta(MU)

Od sada ćemo *multiplicative update* radi jednostavnosti označavati sa kraticom Mu.

Funkcija troška J gore definirana može se zapisati kao

$$\frac{1}{2}||X - UV||_F^2 = \frac{1}{2}\sum_{i=1}^{n}||X_{:i} - UV_{:i}||_2^2$$

dakle može se rastaviti na n nezavisnih problema gdje minimiziramo svaki stupac v od V posebno. Tada imamo niz kvadratnih problema koje možemo izraziti kao $\min_{v\geq 0} F(v)$, gdje je $F(v) = \frac{1}{2}||x - Uv||_2^2$.

Teorem 2.2.1. Euklidska udaljenost $||X - UV||$ je nerastuća primjenom Mu.

Pretpostavimo da je $\overline{h} \geq 0$ neka trenutna aproksimacija NMF problema. Tada možemo formulirati sljedeći problem:

$$\min_{v \geq 0} \overline{F}(v) = \min_{v \geq 0} \frac{1}{2}[||x - Uv||_2^2 + (v - \overline{v})^T W_{\overline{v}}(v - \overline{v})]$$

gdje smo definirali $W_{\overline{v}} = D_y - V^T V$ za $y = \frac{[V^T V \overline{v}]}{[\overline{v}]}$.

Matrica $W_{\overline{v}}$ je pozitivno semidefinitna, stoga je drugi pribrojnik $(v - \overline{v})^T W_{\overline{v}}(v - \overline{v}) \geq 0$, iz čega slijedi

da je $\overline{F}(\overline{v}) = F(\overline{v})$. Raspišemo li gornji izraz, dobivamo da je gradijent od $\overline{F}(v)$ jednak

$$\nabla_v \overline{F} = U^T U v - U^T x + W_{\overline{v}}(v - \overline{v})$$

Izjednačavanjem $\nabla_v \overline{F} = 0$ kako bismo dobili minimum v^* dobijemo

$$(U^T U + W_{\overline{v}})v^* = U^T x - W_{\overline{v}} \overline{v}$$

Kako vrijedi $U^T U + W_{\overline{v}} = D_{U^T U \overline{v}} D_{\overline{v}}^{-1}$ i $V_{\overline{v}} \overline{v} = 0$, konačno imamo:

$$w^* = \overline{w} \circ \frac{[U^T x]}{[U^T U \overline{w}]}$$

gdje $\frac{[]}{[]}$ označava dijeljenje matrica po komponentama. S obzirom da je v^* globalni minimum za $\overline{F}(v)$, vrijedi $\overline{F}(v^*) \leq \overline{F}(v)$. Nadalje, već smo vidjeli da vrijedi $\overline{F}(v) \geq F(v)$ za svaki v.
Sve skupa dokazali smo monotonost.
Ovdje imamo izvod Mu za jedan stupac matrice V. Ako isti postupak primijenimo za svaki stupac matrice, tada dobijemo Mu za cijelu matricu V. Analogno se izvodi za matricu U. □

Činjenica da greška očito svoj minimum poprima u 0 i svojstvo da je nerastuća nam garantiraju konvergenciju u neki L realan broj.
U kasnijim poglavljima ćemo vidjeti kako se ovaj algoritam primjenjuje u praksi, te koja su njegova ograničenja.

3 Klasteriranje tekstualnih podataka

Objasniti kako klasterirati tekstualne podatke koristeći NMF će nam biti glavni zadatak u ovome radu. Kada god se susrećemo sa nekim novim problemom koji želimo rješiti nekom metodom strojnog učenja, vjerojatno i najteži dio samoga zadatka nam je parametrizirati podatke u neki oblik iz kojega možemo primjenom određene metode doći do interpretabilnoga rezultata. Kada to i uspijemo, postavlja se pitanje koliko smo uspješni bili, tj. kakve su nam performanse metode? Bismo li se mogli u nju pouzdati u realnoj situaciji? Možemo li barem naslutiti kada bi nam metoda dala dobre/loše rezultate? Iz kojega razloga? Ova pitanja samo su dio izazova na koje nailazimo pri rješavanju jednoga ovakvoga problema.
Mi ćemo se u ovome radu poslužiti gotovom parametrizacijom. Cilj će nam biti jedan Korpus od 50 artikala podijeliti u kategorije prema tome o čemu se radi u priloženome tekstu.
Za početak ćemo se pozabaviti čišćenjem podataka(data cleaning) i uređivanjem(preporcessing) podataka.

3.1 Priprema podataka za rad

Kako ovaj projekt radimo u programskom jeziku R, onda će i sintaksa biti prikladna tomu.
U programskom jeziku R imamo širok spektar paketa koji nam omogučavaju da vrlo lagano naše tekstualne podatke očistimo. Kada kažemo očistimo mislimo na uklanjanje specijalnih znakova, točke, zareza itd. Izbacujemo sve riječi koje nam ne pridonose značenju, tj. veznike, priloge itd. Na kraju koristimo tzv. word stemmer koji će nam sve riječi sličnog značenja(sinonime) spojiti u jednu riječ. Tu će svakako spadati i razlike poput npr. riječi "imati", "imam" i sl. Na kraju se sve skupi u jednu matricu.

3.2 Parametrizacija

Ovdje ćemo koristiti inačicu široko poznatog $TF - IDF$ standarda za prikaz tekstualnog sadržaja u nekakvom numeričkom obliku. Stupci će nam predstavljati članke, a retci će nam predstavljati riječi iz tih članaka. Formalizirajmo sada prethodno izrečenih par rečenica.

Definiramo prvo $W = \{f_1, f_2, ..., f_m\}$ kompletan vokabular riječi. Ovdje m predstavlja ukupan broj riječi. Definiramo vektor frekvencija X_i za pojedini dokument d_i kao

$$X_i = [x_{1,i}, x_{2,i}, ..., x_{m,i}]^T$$

$$x_{j,i} = t_{j,i} \log(\tfrac{n}{idf_j})$$

gdje su $t_{j,i}, idf_j, n$ frekvencija riječi $f_j \in W$ u dokumentu d_i, broj dokumenata koji sadrži riječ f_j, te ukupan broj dokumenata u Korpusu respektivno.

Vektore X_i ćemo također normirati kako bismo poboljšali performanse algoritama koje koristimo.
Vektore X_i sada koristimo kao stupce, te konstruiramo matricu dimenzija $m \times n$ X. Nazivamo ju *term document matrix*. Na ovoj matrici ćemo primijeniti NMF algoritam.

3.3 Definiranje dodatnih alata

3.3.1 Kriterij zaustavljanja

Prvo ćemo se pozabaviti pojmom kriterija zaustavljanja. Moramo definirati do kada želimo provoditi gornji algoritam Mu.
Algoritam će se iterirati sve dok vrijedi:

$$||U^t - U^{t-1}||_F \geq \epsilon \ \vee \ ||V^t - V^{t-1}||_F \geq \epsilon$$

Za neki malen epsilon.
Ovdje nam eksponenti t i $t-1$ predstavljaju t-tu iteraciju, te njen prethodnik. Dakle algoritam ćemo zaustaviti kada nam se primjenom Mu U i V prestanu značajno mijenjati.

3.4 Postupak evaluiranja rezultata

Glavni problem u realnim situacijama nam je kada nam nisu zadani *text label*, tj. naslovi dokumenata/članaka s kojima radimo. Postavlja se pitanje na koji način biti siguran da je naš algoritam pravilno raspodjelio

članke. Ovdje ćemo izložiti naš prijedlog.
Pretpostavka je da preciznost algoritma asimptotski možemo procijeniti kada više puta klasteriramo iste podatke. Postupak je da uzimamo po 70% uzorka(riječi) svaki puta i provodimo algoritam. Tada te rezultate na 100 iteracija usporedimo i uprosječimo. Ta aritmetička sredina će nam predstavljati efikasnost našeg algoritma na zadanom tekstu.
Upoznajmo se sada sa metodama koje ćemo koristiti za uspoređivanje rezultata.

3.4.1 Općenito

Problem evaluiranja rezultata klasteriranja se može rješiti na mnogo načina. Od načina povezanih sa teorijom grafova(vidi *Kuhn Munkres*) algoritam, do algoritama koji koriste Kombinatornu optimizaciju, sve do nekih Statističkih metoda. Mi ćemo ovdje izložiti jednu metriku, Mutual information, te ćemo ju uz kod objasniti, te usporediti sa drugom metodom $V - measure$.

3.4.2 Mutual information

Mutual information, hrv. Zajednička informacija jedna je od metoda koja na neki način obuhvaća drugu i treću metodu od gore.

Neka su dane dvije particije dokumenata, tj. dvije iteracije klasteriranja A i B. Definiramo zajedničku informaciju $MI(A, B)$ formulom:

$$MI(A,B) = \sum_{a_i \in A, b_j \in B} p(a_i, b_j) \log_2 \frac{p(a_i, b_j)}{p(a_i)p(b_j)}$$

gdje $p(a_i), p(b_j)$ označavaju vjerojatnosti da dokument nasumično odabran iz Korpusa pripada klasteru a_i i b_j respektivno. $p(a_i, b_j)$ predstavlja vjerojatnost da dokument pripada klasterima a_i i b_j. Mjera $MI(A, B)$ poprima vrijednosti između 0 i $\max\{H(A), H(B)\}$, gdje su $H(A)$ i $H(B)$ entropies-i, tj. mjere koje nam govore o razini slučajnosti devijacije u sistemu.
Definiramo ih kao:

$$H(A) = -\sum_{i=1}^{N} \frac{a_i}{N} \log_2(\frac{a_i}{N})$$

$$H(B) = -\sum_{j=1}^{N} \frac{b_j}{N} \log_2(\frac{b_j}{N})$$

Primijetimo kako u skroz općenitoj formulaciji, broj klastera nam ne mora biti nužno jednak.

Ovdje $p(a_i) = \frac{a_i}{N}$, $p(b_j) = \frac{b_j}{N}$, i $p(a_i, b_j) = \frac{n_{i,j}}{n}$ gdje je $n_{i,j}$ #puta kada je prvi model element stavio u i-ti klaster, a drugi model u j-ti klaster.
Zamjenom varijabli kao gore izraz za MI postaje

$$\sum_{a_i \in A, b_j \in B} \frac{n_{i,j}}{N} \log_2(\frac{n_{i,j} N}{a_i b_j})$$

Za kraj je potrebno normirati mjeru tako da ju podijelimo s $\max\{H(A), H(B)\}$.

```
Mutual_info<-function(x,y){
  l<-length(unique(x))
  ai<-numeric(l)
  bj<-numeric(l)
  cont<-table(x,y)
  for(i in 1:l){
    ai[i]<-sum(cont[i,])
    bj[i]<-sum(cont[,i])
  }
  N<-sum(ai)
  H_a<--sum( (ai/N) * log2(ai/N))
  H_b<--sum(   (bj/N)* log2(bj/N))
  MI<-0
  for(i in 1:l){
    for(j in 1:l){
      if(cont[i,j]!=0){
        MI<-MI+(cont[i,j]/N)*log2((cont[i,j]*N)/(ai[i]*bj[j]))
    }
  }
  }
  return(MI/max(H_a,H_b))
}
```

U slučaju da nam je neka frekvencija 0 *definiramo cijeli izraz kao* 0, *tj.* $0\log_2(0) := 0$.

3.4.3 V-mjera

Ovu metodu ćemo koristiti samo za komparaciju rezultata dobivenih metodom zajedničke informacije.

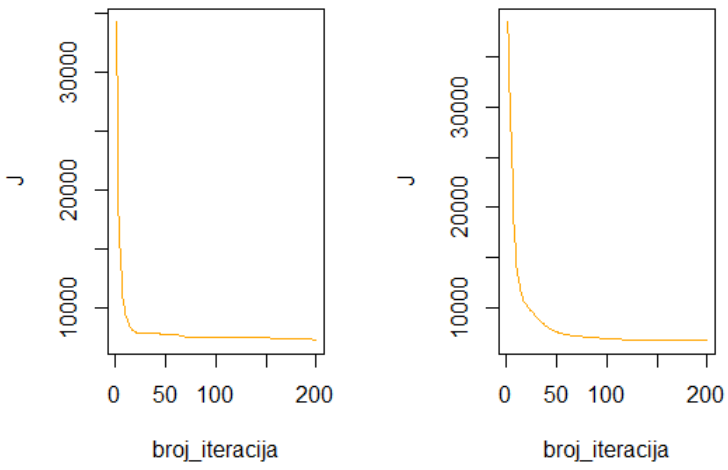
3.5 Transformacija podataka

Pri procesu klasteriranja naišli smo na određene izazove. Najveći od izazova nam je bio sparse/poluprazna ulazna matrica što je i očito uzmemo li u obzir parametrizaciju. Ranije smo već rekli da ulazna matrica X mora biti pozitivna. Dakle sada je jasno da nam trebaju određene transformacije na polaznoj matrici kojima nećemo izgubiti bitne informacije, a postići ćemo pozitivnost. Za taj problem postoje razna rješenja. Od kombiniranja metode sa SVD metodom kao preprocessing do najočitijih transformacija translatiranjem podataka za neku fiksnu vrijednost. Mi smo koristili dvije različite metode. Prva je dodavanjem malenoga epsilona kojim bismo izbjegli pojavljivanje 0 u nazivniku u samome Mu. Druga metoda je jednostavno translatiranje ulaznih vrijednosti za ϵ u desno kako bismo osigurali du sve vrijednosti pozitivne.

3.6 Algoritam NMF i njegove performanse

Sada kada smo pripremili podatke, možemo se posvetiti algoritmu.
Dakle ulazi su nam matrica X i k, ciljani broj klastera. Prvo trebamo definirati polazne vrijednosti za nenegativne matrice $U \in \mathbb{R}^{m\times k}$ i $V \in \mathbb{R}^{k\times n}$. Mi smo to napravili tako da smo ih simulirali iz standardne normalne distribucije, te na kraju uzeli apsolutne vrijednosti dobivenoga. Napomenimo kako je proces definiranja polaznih vrijednosti u praksi vrlo bitan.

Iduči korak je provođenje algoritma Mu za minimiziranje funkcije troška J. Pogledajmo kako se greška ponaša pri prvih 200 iteracija algoritma u oba slučaja.



Možemo zaključiti da prva varijanta, tj. varijanta gdje definiramo $\epsilon > 0$ daje bržu konvergenciju, no neznatno, barem u slučaju ovih podataka. *Napomenimo kako je u praksi brzina konvergencije bitna, posebno u situacijama kada imamo kratak razmak u osvježavanju samoga programa(streaming time).*

Idući cilj nam je iskoristiti algoritam kojim ćemo normalizirati matrice U i V.
Cilj je iterirati sljedeća dva retka

$$u_{i,j} \leftarrow \frac{u_{i,j}}{\sqrt{\sum_i u_{i,j}^2}}$$

$$v_{i,j} \leftarrow v_{i,j} \sqrt{\sum_i u_{i,j}^2}$$

Ovdje smo se odlučili na fiksan broj od 10 iteracija, radi neznatnih promjena.

Za kraj, objasnimo kako metodom dolazimo do klastera.
Koristimo matricu V za to. Svaki redak matrice V, a ima ih k, predstavlja zaseban klaster. Pridružujemo dokument d_i gdje je $i = 1, ..., n$, za n broj dokumenata klasteru x ako je $x = argmax_j[V]_{i,j}$. Ubacimo sada kod za obje inačice.

3.6.1 Prva varijanta NMF

```
NNMF_a<-function(L,k){
  U<-matrix(0,nrow=dim(L)[1],ncol=k)
  V<-t(matrix(0,nrow=k,ncol=dim(L)[2]))
  for(i in 1:k){
    U[,i]<-rnorm(dim(L)[1])
    V[,i]<-rnorm(dim(L)[2])
  }
  U<-abs(U)
  V<-abs(V)
  #algoritam slozenosti O(t*k*n)
  eps=0.01
  U_l<-U
  V_l<-V
  while(1){
    U_l=U
    V_l=V
    U<-U * (L %*% V)/(U %*% t(V) %*% V+eps)
    V<-V * ((t(L) %*% U)/(V %*% t(U) %*% U+eps))
    if(norm(U-U_l)<eps & norm(V-V_l)<eps ){break}
  }
  #normaliziramo U i V
  for(t in 1:10){
    U<-U/sqrt(sum(U*U))
    V<-V/sqrt(  sum(U*U)  )
  }
  klas<-numeric(dim(L)[2])
  z<-apply(t(V),2,max)
  for(i in 1:dim(L)[2]){
    klas[i]<-which(V[i,]==z[i])
  }
  return (klas)
}
```

Primijetimo kako su kriteriji zaustavljanja implementirani u Mu algoritmu.

3.6.2 Druga varijanta NMF

```
NNMF_b<-function(L,k){
  U<-matrix(0,nrow=dim(L)[1],ncol=k)
  V<-t(matrix(0,nrow=k,ncol=dim(L)[2]))
  for(i in 1:k){
    U[,i]<-rnorm(dim(L)[1])
    V[,i]<-rnorm(dim(L)[2])
  }
  U<-abs(U)
  V<-abs(V)
  eps=0.01
  U_l<-U
  V_l<-V
  while(1){
    U<-U_l
    V<-V_l
    if(length(which( (U %*% t(V) %*% V)==0))>0){
      return (0)
    }
    U<-U * (L %*% V)/(U %*% t(V) %*% V)
    V<-V * ((t(L) %*% U)/(V %*% t(U) %*% U))
    if( norm(U-U_l)<eps & norm(V-V_l)<eps ){break}
    U_l<-U
    V_l<-V
  }
  for(t in 1:10){
    U<-U/sqrt(sum(U*U))
    V<-V/sqrt(sum(U*U))
  }
  klas<-numeric(dim(L)[2])
  z<-apply(t(V),2,max)
  for(i in 1:dim(L)[2]){
    klas[i]<-which(V[i,]==z[i])
  }
  return (klas)
}
```

Ovdje je također korišten ϵ, no samo u kriteriju zaustavljanja, ne i u transformaciji.

3.7 Usporedna metoda

Metoda s kojom ćemo uspoređivati dobivene rezultate je metoda k-sredina. Ona predstavlja najosnovniji primjer metode nenadziranoga učenja.

3.8 Rezultati

Pogledajmo sada i usporedimo uspješnost svakoga od gore navedenih algoritama.

k	NMF_tr1	NMF_tr2	K-means
5	0.746 / 0.772	0.280 / 0.282	0.791 / 0.791
10	0.780 / 0.811	0.450 / 0.451	0.906 / 0.906
15	0.810 / 0.840	0.601 / 0.601	0.847 / 0.891
20	0.785 / 0.814	0.701 / 0.701	0.861 / 0.880
25	0.780 / 0.807	0.793 / 0.793	0.809 / 0.855
30	0.835 / 0.851	0.857 / 0.858	0.890 / 0.898
35	0.883 / 0.891	0.876 / 0.876	0.969 / 0.960
40	0.924 / 0.928	0.9205 / 0.9205	0.95 / 0.962

Zaključujemo kako prva transformacija, tj. transformacija dobivena dodavanjem maloga broja epsilon na sve podatke s ciljem postizanja pozitivnosti ulazne matrice za manje k daje bolje rezultate nego druga transformacija. Sve skupa, algoritam k-sredina i dalje daje najbolje rezultate.

Napomena 3.7.1 Pokazuje se da inicijaliziranjem matrica U i V pomoću SVD dekompozicije dolazimo vrlo blizu globalnome minimumu.

Literatura

[1] Wei Xu, Xin Liu and Yihong Gong. *Document Clustering Based On Non-negative Matrix Factorization.*

[2] Dajana Jerončić. *Nenegativne Matrične faktorizacije* . Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Matematički odsjek, 2020.

[3] Mutual Information,
`https://en.wikipedia.org/wiki/Mutual_information`