Singulaariarvohajotelma ja sen sovelluksia data-analytiikassa ja koneoppimisessa

Tarmo Ilves

$$
\begin{aligned}
A_{m\times n} &= U_{m\times m}\Sigma_{m\times n}V^T_{n\times n} \\
&= \begin{bmatrix} | & & | \\ u_1 & \cdots & u_m \\ | & & | \end{bmatrix}
\left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & O \\ 0 & \cdots & \sigma_r & \\ \hline & O & & O \end{array}\right]
\begin{bmatrix} — & v_1^T & — \\ & \vdots & \\ — & v_n^T & — \end{bmatrix}
\end{aligned}
$$

Matematiikan pro gradu

Jyväskylän yliopisto
Matematiikan ja tilastotieteen laitos
Kevät 2024

Tiivistelmä: Tarmo Ilves, *Singulaariarvohajotelma ja sen sovelluksia data-analytiikassa ja koneoppimisessa* (engl. *Singular Value Decomposition and Its Applications in Data Analytics and Machine Learning*), matematiikan pro gradu -tutkielma, 53 sivua, Jyväskylän yliopisto, Matematiikan ja tilastotieteen laitos, kevät 2024.

Tässä tutkielmassa perehdytään singulaariarvohajotelmaan sekä sen hyödyntämiseen data-analytiikan ja koneoppimisen näkökulmasta. Singulaariarvohajotelma on olemassa mille tahansa matriisille A muodossa $A = U\Sigma V^T$, missä U ja V ovat ortonormaaleja matriiseja, ja Σ on diagonaalimatriisi. Matriisin Σ diagonaalialkioita kutsutaan matriisin A singulaariarvoiksi, ja ne on järjestetty suuruudeltaan laskevaan järjestykseen.

Singulaariarvohajotelma ja singulaariarvot mahdollistavat erinomaisen menetelmän alkuperäisen matriisin approksimoimiseksi. Matriisilla on aina astettaan r vastaava määrä singulaariarvoja, ja valitsemalla näistä vain $k < r$ suurinta ja asettamalla loput nolliksi saadaan Eckartin ja Youngin lauseen nojalla paras astetta k oleva approksimaatio alkuperäisestä matriisista. Alempiasteisen matriisiapproksimaation hyödyntäminen on laskentatehokkuuden lisäksi myös datan yksinkertaistamisen kannalta houkuttelevaa, etenkin kun kyseessä on approksimaatioista paras.

Visuaalisin esimerkki parhaasta approksimaatiosta alempiasteisella matriisilla ilmenee tarkastelemalla digitaalisia valokuvia. Digikuvat voidaan esittää matriisimuodossa, mikä mahdollistaa singulaariarvohajotelman käytön. Suurimmat singulaariarvot sisältävät pääpiirteet alkuperäisestä kuvasta, ja pienimmät unohdettaessa saadaan alkuperäistä kuvaa muistuttava approksimaatio, joka vie vähemmän tallennustilaa riippuen valittujen singulaariarvojen määrästä. Valittava k vaikuttaa tallennustilan lisäksi kuvanlaatuun.

Kuvanpakkauksen lisäksi singulaariarvohajotelmaa voidaan soveltaa digitaalisissa palveluissa kerättävän tiedon analysoimiseen, jolloin pystytään tuottamaan käyttäjille personoituja suosituksia. Suosittelujärjestelmien perusideana on tarjota mahdollisimman hyviä suosituksia käyttäjän toiminnan, kuten tuotearvostelujen perusteella. Ihmisten tekemiä arvosteluja esimerkiksi elokuvista voidaan käsitellä suurena datamatriisina, jolloin singulaariarvohajotelmaa on mahdollista käyttää.

Suosittelujärjestelmän rakentamisessa puhutaan yleisesti minimointiongelmasta, jossa halutaan etsiä lähimpänä alkuperäistä datamatriisia R oleva matriisi XY, missä X kuvastaa käyttäjäpiirteitä ja Y tässä tapauksessa elokuviin liittyviä piirteitä. Koska kaikki ihmiset eivät arvostele kaikkia elokuvia, täytyy matriisin tyhjät alkiot ensin alustaa, esimerkiksi käyttäjäkohtaisilla keskiarvoilla. Alustamisella on paljon vaikutusta singulaariarvohajotelmalla saataviin approksimaatioihin ja elokuvasuosituksiin. Tyypillisesti suositusten toimivuutta testataan mallin koulutus- eli opetusjoukosta erillisellä testijoukolla, jota ei ole käytetty approksimaation tekemiseen. Approksimaation tarkkuutta voi parantaa lisäämällä alkuperäiseen minimointiongelmaan regularisointitermin, jolloin paras approksimaatio saadaan vähentämällä singulaariarvoista regularisointikerroin $\gamma \geq 0$. Toinen numeerinen tapa on iteroimalla laskea singulaariarvohajotelma useaan kertaan, ja päivittää ainoastaan puuttuneet arvot kullakin iterointikierroksella saatavilla uusilla approksimaatioilla.

Sisällys

Johdanto

Digitalisaation aikakausi on tullut jäädäkseen. Valtavia datamääriä kerätään jatkuvasti, ja data-analytiikka on osoittautunut entistä tärkeämmäksi datan ymmärtämisen ja hyödyntämisen kannalta. Datan esittäminen matriisimuodossa on usein mahdollista ja hyödyllistä, sillä se avaa lineaarialgebran ja matriisilaskennan koneiston käyttöön. Tässä pro gradu -tutkielmassa käsiteltävä singulaariarvohajotelma tarjoaa tavan erotella isosta datamatriisista keskeisimmät ominaisuudet eli pääpiirteet vähemmän relevantista informaatiosta.

Matriisin A singulaariarvohajotelma on muotoa $A = U\Sigma V^T$, missä matriisit U ja V ovat ortonormaaleja matriiseja, ja Σ diagonaalimatriisi. Matriisin Σ diagonaalialkioita kutsutaan singulaariarvoiksi, ja nämä singulaariarvot sisältävät keskeistä tietoa alkuperäisen matriisin rakenteesta. Singulaariarvot ovat aina positiivisia ja järjestetty diagonaalilla laskevaan järjestykseen, jolloin suurimmat arvot paljastavat datan tärkeimmät ominaisuudet. Carl Eckart ja Gale Young osoittivat 1930-luvulla, että asettamalla pienempiä singulaariarvoja nolliksi saadaan alkuperäistä matriisia approksimoiva alempiasteinen matriisi, joka on approksimaationa paras mahdollinen eli lähimpänä alkuperäistä. Tätä neronleimausta on lähdetty myöhemmin soveltamaan erityisesti datankäsittelyssä, ja mahdollisuuksia on monia.

Luvussa 1 esitellään myöhempien lukujen kannalta tärkeitä lineaarialgebran ja matriisilaskennan määritelmiä ja lauseita. Koko tutkielman läpi tullaan tarvitsemaan käsitteitä esimerkiksi matriisin asteesta sekä singulaariarvoista. Singulaariarvohajotelman olemassaolo mille tahansa matriisille osoitetaan luvussa 2, ja sen jälkeen annetaan myös laskuesimerkki yksinkertaisen matriisin tapauksessa.

Eckartin ja Youngin lause todistetaan luvussa 3, mikä on singulaariarvohajotelman käytön kannalta ehkä merkittävin tulos. Matriisiapproksimaation paremmuuden mittarina käytetään matriisin Frobenius-normia, samaan tyyliin kuin kahden vektorin etäisyyttä mitataan euklidisen normin avulla. Myöhemmin havaitaan, että alempiasteinen matriisiapproksimaatio on hyödyllinen datan yksinkertaistamisen lisäksi myös laskentatehokkuuden kannalta.

Datan pakkaaminen pienempään kokoon säästää tallennustilaa, ja luvussa 4 esitetään tapa pakata digitaalisia valokuvia singulaariarvohajotelman avulla. Ensin käsitellään lyhyesti digikuvien yleistä termistöä ja kuvien esittämistä matriiseina. Kuvanpakkaus singulaariarvohajotelmalla on toteutettu yksinkertaisena Python-ohjelmana, ja esimerkkinä on kokeiltu pakata samaa värivalokuvaa eri singulaariarvojen määrillä.

Luku 5 perehdyttää lukijan suosittelujärjestelmien maailmaan. Esimerkkinä käytetään ihmisten vuorovaikutusta elokuvien kanssa elokuva-arvostelujen muodossa. Suosittelujärjestelmää rakennettaessa halutaan tarjota käyttäjille mahdollisimman hyviä ja henkilökohtaisia suosituksia palvelun tuotteista. Kun data esitetään matriisina, tavoitteena on löytää paras mahdollinen approksimaatio, jolloin singulaariarvohajotelma nousee keskeiseen rooliin. Elokuva-arvostelujen tapauksessa alkuperäinen datamatriisi sisältää paljon tyhjiä alkioita, sillä kaikki ihmiset eivät arvostele kaikkia elokuvia. Matriisin täydentämismenetelmiä käydään aluksi läpi, ja toimivuuksia verrataan pienen esimerkkimatriisin avulla. Saatavat suositukset voivat riippua paljon matriisin alustusvaiheesta, ja approksimaation parantamiseen annetaan myös kaksi muuta lähestymistapaa. Vähentämällä kaikista singulaariarvoista jokin regularisointikerroin saadaan tasattua singulaariarvojen merkitystä, ja iteroimalla tyhjien alkioiden alkuarvauksia singulaariarvohajotelman antamilla uusilla approksimaatioilla on myös mahdollista päästä parempiin suosituksiin. Lopuksi näytetään eri menetelmien toimivuus oikealla MovieLens-elokuva-arvosteludatalla. Tätä varten kirjoitettu Python-ohjelma elokuvasuositusten tekemiseen, sekä kuvanpakkauksessa käytetty ohjelma löytyvät liitteestä A.

LUKU 1

Esitietoja

Käydään aluksi läpi lineaarialgebran kannalta olennaisia asioita, joiden pohjalta teoriaa lähdetään viemään eteenpäin. Lukijan oletetaan tuntevan jo valmiiksi lineaarialgebran ja matriisilaskennan perusteita ja käsitteitä, ja tarvittaessa näitä voi kerrata esimerkiksi David Poolen kirjasta [**14**].

Tässä työssä pysytellään reaalilukujen joukossa $\mathbb{R}$, mutta teorian tasolla monet käytävät asiat on mahdollista yleistää myös kompleksiluvuille. Aloitetaan teoriaosuus viidellä matriiseihin liittyvällä käsitteellä ja merkintätavalla.

MÄÄRITELMÄ 1.1. Olkoon A $m \times n$-matriisi ja B $m \times m$-matriisi.

(i) Matriisin A *sarakeavaruus* on sen sarakevektoreiden virittämä avaruus
$$\operatorname{col}(A) = \{b \in \mathbb{R}^m \colon b = Ax \text{ jollain } x \in \mathbb{R}^n\}.$$

(ii) Matriisin A *riviavaruus* on sen rivivektoreiden virittämä avaruus
$$\operatorname{row}(A) = \operatorname{col}\big(A^T\big).$$

(iii) Matriisin A *ydin* on
$$\operatorname{N}(A) = \{x \in \mathbb{R}^n \colon Ax = 0\}.$$

(iv) Matriisin A *aste* on sen sarakeavaruuden kannan alkioiden lukumäärä eli sarakeavaruuden dimensio
$$\operatorname{rank}(A) = \dim \operatorname{col}(A).$$

(v) Neliömatriisin B *jälki* on sen diagonaalialkioiden summa
$$\operatorname{tr}(B) = \sum_{i=1}^{m} b_{ii}.$$

Joissakin lähteissä sarakeavaruutta kutsutaan kuva-avaruudeksi lineaarikuvausten yhteydessä. Matriisin aste vastaa käytännössä sen lineaarisesti riippumattomien sarakkeiden lukumäärää. Itse asiassa seuraava astelause kertoo, että aste saadaan myös lineaarisesti riippumattomien rivien lukumäärästä.

LAUSE 1.2 (Astelause). *Olkoon A $m \times n$-matriisi. Tällöin matriisin sarakeavaruudella ja riviavaruudella on sama dimensio*
$$\dim \operatorname{row}(A) = \dim \operatorname{col}(A) = \operatorname{rank}(A).$$

TODISTUS. Katso [**12**, lause 3.6.6]. □

Koska matriisin sarakeavaruus on sama kuin saman matriisin transpoosin riviavaruus, saadaan astelauseesta seurauksena todettua matriisien A ja A^T asteiden olevan samat.

SEURAUS 1.3. *Jos A on matriisi, niin*

$$\operatorname{rank}\big(A^T\big) = \operatorname{rank}(A).$$

TODISTUS. Asteen ja riviavaruuden määritelmien sekä astelauseen 1.2 nojalla

$$\begin{aligned}\operatorname{rank}\big(A^T\big) &= \dim \operatorname{col}\big(A^T\big)\\ &= \dim \operatorname{row}(A)\\ &= \operatorname{rank}(A),\end{aligned}$$

kuten haluttiinkin. □

Astelauseen lisäksi toinen hyödyllinen tieto tulee seuraavasta dimensiolauseesta. Sen mukaan matriisin ytimen dimensio ja aste riippuvat toisistaan.

LAUSE 1.4 (Dimensiolause). *Olkoon A $m \times n$-matriisi. Tällöin matriisin A asteelle ja ytimen dimensiolle pätee*

$$\dim \mathrm{N}(A) + \operatorname{rank}(A) = n.$$

TODISTUS. Katso esimerkiksi [**14**, lause 3.26]. □

Tutkitaan seuraavaksi tulomatriisin A^TA ominaisuuksia, sillä myöhemmin kyseisiä ominaisuuksia tullaan hyödyntämään singulaariarvohajotelman muodostamisessa. Dimensiolauseesta saadaan seurauksena yhtäsuuruus matriisien A ja A^TA asteille.

SEURAUS 1.5. *Olkoon A $m \times n$-matriisi. Tällöin*

$$\operatorname{rank}\big(A^TA\big) = \operatorname{rank}(A).$$

TODISTUS. Tarvittaessa katso [**14**, lause 3.28]. □

Asteiden yhtäsuuruuden lisäksi on hyödyllistä todeta matriisin A^TA olevan aina symmetrinen. Symmetrisyyden lisäksi huomataan helposti, että ominaisarvot ovat positiivisia lukuja.

LAUSE 1.6. *Olkoon A $m \times n$-matriisi. Tällöin matriisi A^TA on symmetrinen, sekä sen ominaisarvot ovat ei-negatiivisia.*

TODISTUS. Osoitetaan aluksi matriisin A^TA symmetrisyys. Yksinkertaisesti transpoosin laskusääntöjen nojalla

$$\big(A^TA\big)^T = A^T\big(A^T\big)^T = A^TA.$$

Näytetään sitten ominaisarvojen ei-negatiivisuus. Olkoon λ matriisin A^TA ominaisarvo ja v sitä vastaava ominaisvektori. Lähdettäessä vektorinormin neliöstä liikkeelle saadaan

$$\begin{aligned}\|Av\|^2 &= (Av)^T(Av)\\ &= v^TA^T(Av)\\ &= v^T\big(A^TA\big)v\\ &= v^T\lambda v\\ &= \lambda v^Tv\\ &= \lambda\|v\|^2.\end{aligned}$$

Tästä yhtäsuuruudesta jakamalla vektorin v normin neliöllä saadaan

$$\lambda = \frac{\|Av\|^2}{\|v\|^2} \geq 0,$$

sillä sekä $\|Av\|^2$ että $\|v\|^2$ ovat ei-negatiivisia. Siispä ominaisarvo λ on ei-negatiivinen, eli yleisemmin ilmaistuna matriisin A^TA ominaisarvot ovat ei-negatiivisia, kuten haluttiin. □

Todetaan sitten matriisien A ja A^TA ytimien välinen yhteys. Myös tätä tietoa tarvitaan myöhemmin singulaariarvohajotelmaa käsiteltäessä.

Lause 1.7. *Olkoon A $m \times n$-matriisi. Tällöin matriiseilla A ja A^TA on sama ydin, eli*

$$\mathrm{N}(A) = \mathrm{N}(A^TA).$$

Todistus. Osoitetaan aluksi, että $\mathrm{N}(A) \subset \mathrm{N}(A^TA)$. Olkoon vektori $x \in \mathrm{N}(A)$. Tällöin ytimen määritelmän nojalla $Ax = 0$, josta kertomalla vasemmalta matriisin A transpoosilla saadaan $A^TAx = 0$. Tämä tarkoittaa, että myös $x \in \mathrm{N}(A^TA)$, joten $\mathrm{N}(A) \subset \mathrm{N}(A^TA)$.

Osoitetaan vielä, että $\mathrm{N}(A) \supset \mathrm{N}(A^TA)$. Olkoon vektori $y \in \mathrm{N}(A^TA)$, josta saadaan $A^TAy = 0$. Tarkastelemalla vektorinormin neliötä huomataan, että

$$\|Ay\|^2 = (Ay)^TAy = y^TA^TAy = y^T \cdot 0 = 0.$$

Koska $\|Ay\|^2 = 0$, niin myös $Ay = 0$ ja näin ollen $y \in \mathrm{N}(A)$. Siis $\mathrm{N}(A) \supset \mathrm{N}(A^TA)$, joten $\mathrm{N}(A) = \mathrm{N}(A^TA)$. □

Matriisin A^TA ominaisuuksien lisäksi täytyy vielä esitellä aliavaruuksiin liittyvä lause, jossa nähdään aliavaruuden U ja sen ortogonaalikomplementin $U^\perp$ välinen yhteys.

Lause 1.8. *Jos U on avaruuden $\mathbb{R}^n$ aliavaruus, niin tällöin*

$$\dim U + \dim U^\perp = n.$$

Lisäksi, jos vektorit $u_1, \ldots, u_r$ muodostavat avaruuden U kannan ja vastaavasti vektorit $u_{r+1}, \ldots, u_n$ muodostavat kannan ortogonaalikomplementille $U^\perp$, niin näiden yhdistetty joukko $\{u_1, \ldots, u_r, u_{r+1}, \ldots, u_n\}$ on avaruuden $\mathbb{R}^n$ kanta.

TODISTUS. Katso [**12**, lause 5.2.2]. □

Mentäessä kohti singulaariarvohajotelmaa halutaan varmasti tietää paljon eri asioita, mutta yksi ensimmäisistä esiin nousevista kysymyksistä liittyy todennäköisesti hajotelman nimeen: mitä ovat singulaariarvot? Määritellään ne seuraavaksi.

MÄÄRITELMÄ 1.9. Olkoon A $m \times n$-matriisi. Matriisin A *singulaariarvot* ovat matriisin A^TA ominaisarvojen neliöjuuret $\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}$. Merkitään $\sigma_j = \sqrt{\lambda_j}$, kun $j = 1, \ldots, n$, ja järjestetään singulaariarvot laskevaan järjestykseen $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$.

Singulaariarvojen hienous ilmenee siinä, että niitä on kaikentyyppisillä matriiseilla. Ominaisarvot sen sijaan ovat vain neliömatriisien ominaisuus. Tämä onkin oleellinen asia seuraavassa luvussa esiteltävän singulaariarvohajotelman ja sen käytettävyyden kannalta. Ennen singulaariarvohajotelmaan menemistä osoitetaan myöhempiä lukuja varten hyödyllinen *astehajotelma*, joka mahdollistaa matriisin jakamisen kahteen täyttä astetta olevaan matriisiin. Matriisi on täyttä astetta, jos $\operatorname{rank}(A) = \min\{m, n\}$, kun A on $m \times n$-matriisi. Astehajotelma on esitelty kirjassa [**2**, määritelmä 5.3].

LAUSE 1.10 (Astehajotelma). *Olkoon A $m \times n$-matriisi, jolle $\operatorname{rank}(A) = r$. Tällöin matriisilla A on olemassa esitys $A = XY$, missä X on $m \times r$-matriisi, Y on $r \times n$-matriisi ja $\operatorname{rank}(X) = \operatorname{rank}(Y) = r$.*

TODISTUS. Olkoon A $m \times n$-matriisi, ja $\operatorname{rank}(A) = r$. Merkitään

$$A = \begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix}.$$

Nyt astelauseen 1.2 nojalla on matriisilla A sellaiset lineaarisesti riippumattomat sarakkeet $a_{k_1}, \ldots, a_{k_r}$, että

$$a_i = y_{1i}a_{k_1} + \cdots + y_{ri}a_{k_r} = \begin{bmatrix} | & & | \\ a_{k_1} & \cdots & a_{k_r} \\ | & & | \end{bmatrix} \begin{bmatrix} — & y_{1i} & — \\ & \vdots & \\ — & y_{ri} & — \end{bmatrix}$$

aina, kun $i = 1, \ldots, n$. Tästä saadaan

$$A = \begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ a_{k_1} & \cdots & a_{k_r} \\ | & & | \end{bmatrix} \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & \ddots & \vdots \\ y_{r1} & \cdots & y_{rn} \end{bmatrix} := XY.$$

Nyt siis X on $m \times r$-matriisi ja Y on $r \times n$-matriisi. Selvästi $\operatorname{rank}(X) = r$, ja jos olisi $\operatorname{rank}(Y) < r$, niin $r = \operatorname{rank}(A) = \operatorname{rank}(XY) \leq \min\{\operatorname{rank}(X), \operatorname{rank}(Y)\} < r$, mikä on ristiriita. Siispä myös $\operatorname{rank}(Y) = r$, mikä osoittaa väitteen. □

Astehajotelman esitystapaa hyödynnetään luvuissa 3 ja 5. Tutustutaan kuitenkin ensiksi tarkemmin singulaariarvohajotelmaan.

LUKU 2

Singulaariarvohajotelma

Matriisit ja niiden käsittely muodostavat olennaisen osan data-analytiikkaa. Kun työskennellään suurien datamäärien parissa, mahdollisten datamatriisien purkaminen osiin tuntuu loogiselta lähestymistavalta. Tämä onkin singulaariarvohajotelman idea, kuten nimestä voidaan päätellä. Voidaankin sanoa, että lineaarialgebra ja data-analytiikka kohtaavat singulaariarvohajotelmassa. Tämä luku pohjautuu lähteisiin [**12**] ja [**14**]. Tutustutaan nyt kenties yhteen tärkeimmistä matriisihajotelmista, singulaariarvohajotelmaan (engl. Singular Value Decomposition, SVD). Samalla osoitetaan, että singulaariarvohajotelma on olemassa kaikille matriiseille.

LAUSE 2.1 (Singulaariarvohajotelma, SVD). *Olkoon A $m \times n$-matriisi, jonka aidosti positiiviset singulaariarvot ovat $\sigma_1 \geq \cdots \geq \sigma_r > 0$. Tällöin on olemassa ortogonaalinen $m \times m$-matriisi U, ortogonaalinen $n \times n$-matriisi V, sekä $m \times n$-matriisi Σ, joiden avulla matriisi A voidaan hajottaa muotoon*

$$A = U\Sigma V^T = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_m \\ | & & | \end{bmatrix} \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & O \\ 0 & \cdots & \sigma_r & \\ \hline & O & & O \end{array}\right] \begin{bmatrix} — & v_1^T & — \\ & \vdots & \\ — & v_n^T & — \end{bmatrix}.$$

Tässä matriisi U koostuu matriisin AA^T ominaisvektoreista $u_1, \ldots, u_m$. Matriisi V puolestaan muodostuu matriisin A^TA ominaisvektoreista. Matriisin Σ diagonaalialkiot ovat matriisin A aidosti positiiviset singulaariarvot, ja Σ täydennetään $m \times n$-kokoiseksi tilanteen mukaan sopivan kokoisilla nollamatriiseilla O.

TODISTUS. Tämä todistus löytyy myös kirjasta [**12**, lause 6.5.1]. Lauseen 1.6 nojalla A^TA on symmetrinen matriisi, ja sen ominaisarvot ovat ei-negatiivisia. Tällöin matriisille A^TA on olemassa sen ortogonaalisesti diagonalisoiva matriisi V. Oletetaan, että matriisin V sarakevektorit on järjestetty siten, että niitä vastaavat ominaisarvot ovat laskevassa järjestyksessä

$$\lambda_1 \geq \cdots \geq \lambda_n \geq 0.$$

Olkoon r matriisin A aste. Tällöin seurauksen 1.3 nojalla myös matriisin A^TA aste on r. Koska A^TA on symmetrinen, niin lauseen 1.6 nojalla sen positiivisten ominaisarvojen määrä myös r. Tästä seuraa

$$\lambda_1 \geq \cdots \geq \lambda_r > 0 \qquad \text{ja} \qquad \lambda_{r+1} = \cdots = \lambda_n = 0,$$

joten myös singulaariarvoille pätee

$$\sigma_1 \geq \cdots \geq \sigma_r > 0 \qquad \text{ja} \qquad \sigma_{r+1} = \cdots = \sigma_n = 0.$$

Muodostetaan matriisi V ominaisarvoihin liittyvien ominaisvektorien avulla siten, että $V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$, missä

$$V_1 = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_r \\ | & & | \end{bmatrix} \qquad \text{ja} \qquad V_2 = \begin{bmatrix} | & & | \\ v_{r+1} & \cdots & v_n \\ | & & | \end{bmatrix}.$$

Olkoon lisäksi matriisi Σ_1 siten, että

$$\Sigma_1 = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix}.$$

Nyt Σ_1 on diagonaalimatriisi, joka koostuu positiivisista singulaariarvoista. Täydennetään Σ_1 $m \times n$-kokoiseksi sopivan kokoisilla nollamatriiseilla O, jolloin saadaan

$$\Sigma = \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & O \\ 0 & \cdots & \sigma_r & \\ \hline & O & & O \end{array}\right].$$

Matriisi V_2 koostuu niistä matriisin A^TA ominaisvektoreista, joille $\lambda = 0$. Tästä seuraa

$$A^TAv_j = 0 \qquad \text{kaikilla } j = r+1, \ldots, n.$$

Lisäksi kyseiset sarakevektorit muodostavat ortonormaalin kannan ytimelle $\mathrm{N}(A^TA)$. Toisaalta lauseen 1.7 nojalla $\mathrm{N}(A^TA) = \mathrm{N}(A)$, joten myös $AV_2 = O$. Koska V muodostuu ominaisvektoreista, on se ortogonaalinen matriisi. Kun I on yksikkömatriisi, voidaan kirjoittaa

$$\begin{aligned} I = VV^T &= V_1V_1^T + V_2V_2^T, \\ A = AI &= AV_1V_1^T + AV_2V_2^T = AV_1V_1^T. \end{aligned} \tag{2.1}$$

Nyt on muodostettu matriisit V sekä Σ singulaariarvohajotelman vaatimalla tavalla. Muodostetaan lopuksi orgononaalinen $m \times m$-matriisi U siten, että

$$A = U\Sigma V^T.$$

Tämä voidaan kirjoittaa muotoon

$$AV = U\Sigma. \tag{2.2}$$

Olkoot $u_1, \ldots, u_m$ matriisin U sarakevektoreita. Tutkittaessa lauseketta (2.2) huomataan, että

$$Av_j = \sigma_j u_j \qquad \text{kaikilla } j = 1, \ldots, r.$$

Tästä saadaan

$$u_j = \frac{1}{\sigma_j} Av_j \qquad \text{kaikilla } j = 1, \ldots, r. \tag{2.3}$$

Merkitään nyt

$$U_1 = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_r \\ | & & | \end{bmatrix},$$

jolloin saadaan yhtäsuuruus

$$AV_1 = U_1\Sigma_1. \tag{2.4}$$

Matriisin U_1 sarakevektorit ovat ortonormaaleja, koska

$$\begin{aligned} u_i^T u_j &= \left(\frac{1}{\sigma_i} v_i^T A^T\right)\left(\frac{1}{\sigma_j} A v_j\right), \quad 1 \le i \le r, \quad 1 \le j \le r \\ &= \frac{1}{\sigma_i \sigma_j} v_i^T \left(A^T A v_j\right) \\ &= \frac{1}{\sigma_i \sigma_j} v_i^T (\lambda_j v_j) \\ &= \frac{1}{\sigma_i \sigma_j} v_i^T \left(\sigma_j^2 v_j\right) \\ &= \frac{\sigma_j}{\sigma_i} v_i^T v_j \\ &= \begin{cases} 1, & \text{kun } i = j \\ 0, & \text{kun } i \neq j. \end{cases} \end{aligned}$$

Kohdasta (2.3) seuraa, että matriisin U_1 sarakevektorit kuuluvat sarakeavaruuteen $\operatorname{col}(A)$. Koska asteen määritelmän mukaan $\dim \operatorname{col}(A) = \operatorname{rank}(A) = r$, muodostavat vektorit $u_1, \ldots, u_r$ ortonormaalin kannan avaruudelle $\operatorname{col}(A)$. Olkoon $\{u_{r+1}, \ldots, u_m\}$ avaruuden $\operatorname{col}(A)^\perp$ ortonormaali kanta. Merkitään

$$U_2 = \begin{bmatrix} | & & | \\ u_{r+1} & \cdots & u_m \\ | & & | \end{bmatrix},$$

ja täydennetään matriisiksi U

$$U = \begin{bmatrix} U_1 & U_2 \end{bmatrix}.$$

Nyt U muodostaa avaruuden $\mathbb{R}^m$ ortonormaalin kannan lauseen 1.8 nojalla. Näytetään lopuksi haluttu yhtäsuuruus $U\Sigma V^T = A$. Kohtien (2.1) ja (2.4) avulla saadaan

$$\begin{aligned} U\Sigma V^T &= \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & O \\ O & O \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \\ &= U_1 \Sigma_1 V_1^T \\ &= A V_1 V_1^T \\ &= A, \end{aligned}$$

kuten alun perin haluttiinkin osoittaa. □

Singulaariarvohajotelman muodostaminen matriisille voidaan tehdä käsin, mutta käytännössä isompia datamatriiseja pyöriteltäessä annetaan tietokoneen hoitaa laskentatyö. Muodostetaan seuraavaksi yksinkertaiselle matriisille singulaariarvohajotelma vaihe vaiheelta lineaarialgebran oppeja hyödyntäen.

ESIMERKKI 2.2. Määritetään singulaariarvohajotelma matriisille $A = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$.

Lasketaan aluksi

$$A^TA = \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -1\cdot(-1)+0\cdot 0 & -1\cdot 0+0\cdot 1 & -1\cdot(-1)+0\cdot 0 \\ 0\cdot(-1)+1\cdot 0 & 0\cdot 0+1\cdot 1 & 0\cdot(-1)+1\cdot 0 \\ -1\cdot(-1)+0\cdot 0 & -1\cdot 0+0\cdot 1 & -1\cdot(-1)+0\cdot 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Ratkaistaan seuraavaksi matriisin A^TA ominaisarvot karakteristisen polynomin avulla. Olkoon λ matrisiin A^TA ominaisarvo, jolloin

$$\begin{aligned} \det\big(A^TA - \lambda I\big) &= \begin{vmatrix} 1-\lambda & 0 & 1 \\ 0 & 1-\lambda & 0 \\ 1 & 0 & 1-\lambda \end{vmatrix} \\ &= (1-\lambda)\begin{vmatrix} 1-\lambda & 0 \\ 0 & 1-\lambda \end{vmatrix} - 0\begin{vmatrix} 0 & 0 \\ 1 & 1-\lambda \end{vmatrix} + \begin{vmatrix} 0 & 1-\lambda \\ 1 & 0 \end{vmatrix} \\ &= (1-\lambda)(1-\lambda)(1-\lambda) - (1-\lambda) \\ &= (1-\lambda)\big((1-\lambda)^2 - 1\big) \\ &= (1-\lambda)\big(\lambda^2 - 2\lambda\big) \\ &= -\lambda(\lambda-1)(\lambda-2). \end{aligned}$$

Nyt karakteristisen polynomin nollakohdat eli juuret ovat matriisin A^TA ominaisarvot, joten

$$\begin{aligned} & -\lambda(\lambda-1)(\lambda-2) = 0 \\ \iff \quad & \lambda = 0, \quad \lambda = 1 \quad \text{tai} \quad \lambda = 2. \end{aligned}$$

Suuruusjärjestyksessä ominaisarvot ovat siis $\lambda_1 = 2$, $\lambda_2 = 1$ ja $\lambda_3 = 0$. Muodostetaan nyt ominaisarvoja vastaavat ominaisvektorit. Esimerkiksi ominaisarvolle $\lambda_1 = 2$:

$$\begin{aligned} & \begin{bmatrix} 1-2 & 0 & 1 \\ 0 & 1-2 & 0 \\ 1 & 0 & 1-2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ \iff \quad & \begin{cases} -x_1 + x_3 = 0 \\ -x_2 = 0 \\ x_1 - x_3 = 0 \end{cases} \\ \iff \quad & \begin{cases} x_1 = x_3 \\ x_2 = 0 \\ x_3 \in \mathbb{R}. \end{cases} \end{aligned}$$

Nyt voidaan valita esimerkiksi $x_3 = 1$, jolloin ominaisarvoa λ_1 vastaavaksi ominaisvektoriksi muodostuu

$$w_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

Vastaavalla tavalla saadaan ominaisarvoja $\lambda_2 = 1$ ja $\lambda_3 = 0$ vastaavat ominaisvektorit, joiksi kelpaavat esimerkiksi

$$w_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{ja} \quad w_3 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}.$$

Helposti nähdään, että ominaisvektorit ovat ortogonaalisia keskenään. Matriisin V muodostamista varten nämä tulee kuitenkin vielä normeerata, eli kunkin vektorin pituuden tulee olla 1. Ortonormaaleiksi ominaisvektoreiksi saadaan

$$v_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{ja} \quad v_3 = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Näiden avulla saadaan matriisiksi V

$$V = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix},$$

ja singulaariarvohajotelmassa käytettäväksi matriisin V transpoosiksi

$$V^T = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Tarvittavat singulaariarvot saadaan laskemalla matriisin $A^T A$ ominaisarvojen neliöjuuret, joten

$$\sigma_1 = \sqrt{2}, \quad \sigma_2 = 1 \quad \text{ja} \quad \sigma_3 = 0.$$

Nyt Σ-matriisin muodostamisessa hyödynnetään ainoastaan aidosti positiivset singulaariarvot, ja täydentämällä matriisi oikean kokoiseksi saadaan

$$\Sigma = \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Muodostetaan vielä matriisi U. Tarvittavat sarakevektorit saadaan laskemalla lausekkeen (2.3) mukaisesti

$$u_1 = \frac{1}{\sigma_1} A v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \cdot \left(\frac{1}{\sqrt{2}}\right) + 0 \cdot 0 + (-1) \cdot \left(\frac{1}{\sqrt{2}}\right) \\ 0 \cdot \left(\frac{1}{\sqrt{2}}\right) + 1 \cdot 0 + 0 \cdot \left(\frac{1}{\sqrt{2}}\right) \end{bmatrix}$$
$$= \begin{bmatrix} -1 \\ 0 \end{bmatrix},$$

$$u_2 = \frac{1}{\sigma_2} A v_2 = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Vektoria u_3 ei ole olemassa, ja lisäksi $\sigma_3 = 0$. Matriisin U muodostamiseksi kuitenkin riittää vektorit u_1 ja u_2, koska nolla-singulaariarvot jätettiin muutenkin huomiotta. Siten matriisi U on

$$U = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Nyt on saatu muodostettua singulaariarvohajotelmaan tarvittavat kolme matriisia, ja

$$A = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} = U\Sigma V^T.$$

Laskemalla voi tarvittaessa tarkistaa, että hajotelma pätee.

Laskuesimerkistä voidaan tehdä havainto, että vaikka singulaariarvot ovat aina yksikäsitteisiä ja siten muodostettava matriisi Σ on yksikäsitteinen, niin matriisit U ja V riippuvat valittavista ominaisvektoreista ja siten eivät ole yksikäsitteisiä. Siispä matriisin singulaariarvohajotelma ei ole yksikäsitteinen.

Hajotelman matriisin U vektoreita $u_1, \ldots, u_m$ kutsutaan *vasemmanpuoleisiksi singulaarivektoreiksi*, ja matriisin V vektoreita $v_1, \ldots, v_n$ kutsutaan *oikeanpuoleisiksi singulaarivektoreiksi*. Singulaariarvohajotelman Σ-matriisi voi sisältää paljon nolla-alkioita, jos alkuperäisen matriisin A aste on pieni ja siten ydin suuri. Nämä nolla-alkiot eivät itse asiassa vaikuta matriisituloon millään tavalla, ja singulaariarvohajotelma saadaan yksinkertaistettua muotoon

$$\begin{aligned}
A = U\Sigma V^T &= \left[\begin{array}{ccc} | & & | \\ u_1 & \cdots & u_m \\ | & & | \end{array}\right] \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & O \\ 0 & \cdots & \sigma_r & \\ \hline & O & & O \end{array}\right] \left[\begin{array}{ccc} — & v_1^T & — \\ & \vdots & \\ — & v_n^T & — \end{array}\right] \\
&= \left[\begin{array}{ccccccc} | & & | & | & & | \\ u_1 & \cdots & u_r & u_{r+1} & \cdots & u_m \\ | & & | & | & & | \end{array}\right] \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & O \\ 0 & \cdots & \sigma_r & \\ \hline & O & & O \end{array}\right] \left[\begin{array}{ccc} — & v_1^T & — \\ & \vdots & \\ — & v_r^T & — \\ — & v_{r+1}^T & — \\ & \vdots & \\ — & v_n^T & — \end{array}\right] \\
&= \left[\begin{array}{ccc} | & & | \\ u_1 & \cdots & u_r \\ | & & | \end{array}\right] \left[\begin{array}{ccc} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{array}\right] \left[\begin{array}{ccc} — & v_1^T & — \\ & \vdots & \\ — & v_r^T & — \end{array}\right] \\
&\quad + \left[\begin{array}{ccc} | & & | \\ u_{r+1} & \cdots & u_m \\ | & & | \end{array}\right] \left[\begin{array}{c} \\ O \\ \\ \end{array}\right] \left[\begin{array}{ccc} — & v_{r+1}^T & — \\ & \vdots & \\ — & v_n^T & — \end{array}\right] \\
&= \left[\begin{array}{ccc} | & & | \\ u_1 & \cdots & u_r \\ | & & | \end{array}\right] \left[\begin{array}{ccc} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{array}\right] \left[\begin{array}{ccc} — & v_1^T & — \\ & \vdots & \\ — & v_r^T & — \end{array}\right] := U_r\Sigma_r V_r^T.
\end{aligned}$$

Tämä muoto singulaariarvohajotelmasta kulkee lähteestä riippuen nimellä *katkaistu SVD* tai *kompakti SVD*. Laskennallisesti tehokkaampaa onkin määrätä ainoastaan r ensimmäistä singulaarivektoria, ja niin useimmiten tietokoneella laskettaessa myös toimitaan. Koska singulaariarvot asetettiin Σ-matriisin diagonaalille laskevassa suuruusjärjestyksessä ja näin ollen viimeiset singulaariarvot voivat mahdollisesti

olla lähelläkin nollaa, herää kysymys voisiko alkuperäisen matriisin hajotelmaa arvioida yksinkertaistamalla sitä lisää, eli esimerkiksi muuttamalla häviävän pieniä singulaariarvoja nolla-alkioiksi. Numeerisessa laskennassa tulee kuitenkin olla tarkkana, sillä pyöristysvirheiden vuoksi singulaariarvo voi näyttää nollasta poikkeavalta, vaikka todellisuudessa olisikin nolla. Seuraavaksi perehdytään matriisin approksimointiin valikoimalla singulaariarvoista vain suurimmat matriisin uudelleenluontivaiheessa.

LUKU 3

Paras approksimaatio

Tämä luku perustuu Gilbert Strangin teokseen [**15**]. Singulaariarvohajotelma mahdollistaa alkuperäisen matriisin arvioimisen alempiasteisilla matriiseilla. Edellisessä luvussa huomattiin, että Σ-matriisin nolladiagonaalialkiot eivät vaikuta alkuperäisen matriisin muodostamiseen. Nollasta poikkeavia singulaariarvoja voidaan myös muuttaa nolla-alkioiksi, jolloin saadaan approksimaatio alkuperäisestä matriisista. Esimerkiksi jos alkuperäinen matriisi A on astetta r, valitaan jokin $k < r$ ja muutetaan singulaariarvot $\sigma_{k+1}, \ldots, \sigma_r$ nolla-alkioiksi, saadaan muodostettua alkuperäistä matriisia approksimoiva matriisi A', joka on astetta k.

Kiinnostavaa olisi tietää, kuinka hyvä tällainen approksimaatio alempiasteisella matriisilla on. Vektoreita vertailtaessa tutkitaan vektorin ja sen approksimaation normia eli etäisyyttä. Sama idea toimii myös matriisien tapauksessa, ja vektorien euklidinen normi voidaan yleistää matriiseille. Ensin tarvitaan kuitenkin matriisien sisätulon määritelmä.

MÄÄRITELMÄ 3.1. Olkoot A ja B $m \times n$-matriiseja. Matriisien A ja B sisätulo on

$$\langle A, B \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij} = \operatorname{tr}\big(B^T A\big).$$

Seuraavaksi esitellään *Frobeniuksen normi*, joka on matriisien sisätulon indusoima normi.

MÄÄRITELMÄ 3.2. Olkoon A $m \times n$-matriisi. Matriisin A *Frobeniuksen normi* on

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2} = \sqrt{\operatorname{tr}(A^T A)}.$$

Frobenius-normi vastaa siis vektorien euklidista normia. Lisäksi Frobenius-normi soveltuu hyvin approksimaation tulkitsemiseen seuraavien tulosten myötä.

LAUSE 3.3. *Jos A on $m \times n$-matriisi ja Q on ortogonaalinen $m \times m$-matriisi, niin tällöin*

$$\|QA\|_F = \|A\|_F .$$

TODISTUS. Koska matriisi Q on ortogonaalinen, niin $Q^TQ = QQ^T = I$. Nyt Frobenius-normin, jäljen ja matriisin Q ortogonaalisuuden nojalla saadaan

$$\begin{aligned}\|QA\|_F^2 &= \sum_{i=1}^{m}\sum_{j=1}^{n}(qa)_{ij}^2 \\ &= \operatorname{tr}\Big((QA)^T QA\Big) \\ &= \operatorname{tr}\big(A^TQ^TQA\big) \\ &= \operatorname{tr}\big(A^TA\big) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}a_{ij}^2 \\ &= \|A\|_F^2 .\end{aligned}$$

Tämä osoittaa halutun yhtäsuuruuden. □

Frobenius-normi ei siis muutu, vaikka alkuperäistä matriisia kerrotaan ortogonaalisella matriisilla. Singulaariarvohajotelmassa matriisit U ja V^T olivat ortogonaalisia, joten tästä saadaan seurauksena hyödyllinen yhteys matriisin Frobenius-normille ja singulaariarvoille.

SEURAUS 3.4. *Olkoon A $m \times n$-matriisi ja olkoot $\sigma_1, \ldots, \sigma_r$ matriisin singulaariarvot. Tällöin*

$$\|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}.$$

TODISTUS. Olkoon matriisin A singulaariarvohajotelma $A = U\Sigma V^T$. Nyt hyödyntämällä lausetta 3.3 saadaan

$$\begin{aligned}\|A\|_F &= \left\|U\Sigma V^T\right\|_F \\ &= \left\|\Sigma V^T\right\|_F \\ &= \left\|\left(\Sigma V^T\right)^T\right\|_F \\ &= \left\|V\Sigma^T\right\|_F \\ &= \left\|\Sigma^T\right\|_F \\ &= \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}.\end{aligned}$$

□

Kun matriisin singulaariarvot ovat tiedossa, saadaan Frobenius-normi laskettua niiden avulla. Toisaalta jos alkuperäisen matriisin pienempiä singulaariarvoja approksimoidaan nolla-alkioiksi, saadaan seurauksen 3.4 avulla verrattua alkuperäistä matriisia tähän alempiasteiseen matriisiin. Itse asiassa tällä tavalla saadaan paras approksimaatio alkuperäiselle matriisille käyttäen alempiasteista matriisia. Jotta kyseinen väite voitaan osoittaa, tarvitaan ensin apulause kertomaan tällaisen parhaan approksimaation olevan olemassa mille tahansa matriisille.

LEMMA 3.5. *Olkoon A $m \times n$-matriisi astetta r, ja olkoon $\mathcal{M}$ kaikkien niiden $m \times n$-matriisien joukko, joiden aste on korkeintaan k ja $0 < k < r$. Tällöin on olemassa matriisi $X \in \mathcal{M}$, jolle*

$$\|A - X\|_F = \min_{S \in \mathcal{M}} \|A - S\|_F .$$

TODISTUS. Muistetaan, että Frobenius-normi on matriisien sisätulon indusoima normi. Sen perusteella olkoon X matriisin A ortogonaaliprojektio joukkoon $\mathcal{M}$. Tällöin, jos $S \in \mathcal{M}$, niin erotusmatriisit $X - S$ ja $A - X$ ovat ortogonaalisia. Nyt Pythagoraan nojalla

$$\|A - S\|_F^2 = \|A - X\|_F^2 + \|X - S\|_F^2 \geq \|A - X\|_F^2 ,$$

joten $\|A - X\|_F = \min_{S \in \mathcal{M}} \|A - S\|_F$. Koska $X \in \mathcal{M}$, niin väite on osoitettu. □

Tämä apulause siis kertoo, että matriisille A löytyy jokin sitä approksimoiva alempiasteinen matriisi, joka on lähimpänä alkuperäistä matriisia. Puhutaan siis parhaasta approksimaatiosta, ja seuraava tulos kertoo singulaariarvohajotelman merkityksen tällaisen parhaan approksimaation etsimisessä.

LAUSE 3.6 (Eckartin ja Youngin lause). *Olkoon A $m \times n$-matriisi, jonka aste on r ja jolla on singulaariarvohajotelma $A = U\Sigma V^T$. Olkoon $\mathcal{M}$ kaikkien niiden $m \times n$-matriisien joukko, joiden aste on korkeintaan k ja $0 < k < r$. Jos $A' = U\Sigma_k V^T$, missä*

$$\Sigma_k = \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & O \\ 0 & \cdots & \sigma_k & \\ \hline & O & & O \end{array}\right],$$

niin tällöin

$$\|A - A'\|_F = \min_{S \in \mathcal{M}} \|A - S\|_F .$$

TODISTUS. Vastaava todistus on esitetty kirjassa [**15**, sivu 74]. Lemman 3.5 nojalla voidaan olettaa, että on olemassa korkeintaan astetta k oleva matriisi X, joka on lähimpänä matriisia A. Halutaan osoittaa, että eräs ratkaisu on $X = A'$. Lauseen 2.1 nojalla matriisille X on olemassa singulaariarvohajotelma

$$X = U \begin{bmatrix} D & O \\ O & O \end{bmatrix} V^T,$$

missä diagonaalimatriisi D on $k \times k$-matriisi singulaariarvoilla $\sigma_1, \ldots, \sigma_k$. Suoraan ei voida sanoa, että diagonalisoivatko tämän hajotelman ortonormaalit matriisit U ja V myös matriisin A. Kirjoitetaan varmuuden vuoksi

$$A = U \begin{bmatrix} L + E + R & F \\ G & H \end{bmatrix} V^T,$$

missä L on alakolmiomatriisi, E diagonaalimatriisi ja R yläkolmiomatriisi siten, että matriisien L ja R diagonaalialkiot ovat nollia ja kukin näistä kolmesta matriisista on kokoa $k \times k$. Matriisit F, G ja H täyttävät koko matriisin $m \times n$-kokoiseksi. Nyt

halutaan näyttää, että L, R, F ja G ovat kaikki nollamatriiseja. Muodostetaan kolmas matriisi $C \in \mathcal{M}$ siten, että

$$C = U \begin{bmatrix} L+D+R & F \\ O & O \end{bmatrix} V^T.$$

Nyt vertaamalla Frobenius-normien neliöitä keskenään saadaan normin laskusääntöjen ja lauseen 3.3 nojalla

$$\begin{aligned}
\|A-X\|_F^2 &= \left\| U \begin{bmatrix} L+E+R & F \\ G & H \end{bmatrix} V^T - U \begin{bmatrix} D & O \\ O & O \end{bmatrix} V^T \right\|_F^2 \\
&= \left\| U \left(\begin{bmatrix} L+E+R & F \\ G & H \end{bmatrix} - \begin{bmatrix} D & O \\ O & O \end{bmatrix} \right) V^T \right\|_F^2 \\
&= \left\| \left(\left(\begin{bmatrix} L+E+R & F \\ G & H \end{bmatrix} - \begin{bmatrix} D & O \\ O & O \end{bmatrix} \right) V^T \right)^T \right\|_F^2 \\
&= \left\| V \left(\begin{bmatrix} L+E+R & F \\ G & H \end{bmatrix} - \begin{bmatrix} D & O \\ O & O \end{bmatrix} \right)^T \right\|_F^2 \\
&= \left\| \begin{bmatrix} L+E+R & F \\ G & H \end{bmatrix} - \begin{bmatrix} D & O \\ O & O \end{bmatrix} \right\|_F^2 \\
&= \left\| \begin{bmatrix} L+E+R & F \\ G & H \end{bmatrix} - \begin{bmatrix} L+D+R & F \\ O & O \end{bmatrix} + \begin{bmatrix} L+R & F \\ O & O \end{bmatrix} \right\|_F^2 \\
&= \left\| \begin{bmatrix} L+E+R & F \\ G & H \end{bmatrix} - \begin{bmatrix} L+D+R & F \\ O & O \end{bmatrix} \right\|_F^2 + \|L\|_F^2 + \|R\|_F^2 + \|F\|_F^2 \\
&= \left\| U \begin{bmatrix} L+E+R & F \\ G & H \end{bmatrix} V^T - U \begin{bmatrix} L+D+R & F \\ O & O \end{bmatrix} V^T \right\|_F^2 + \|L\|_F^2 + \|R\|_F^2 \\
&\quad + \|F\|_F^2 \\
&= \|A-C\|_F^2 + \|L\|_F^2 + \|R\|_F^2 + \|F\|_F^2 .
\end{aligned}$$

Koska X oli lähimpänä matriisia A, niin etäisyys $\|A-X\|_F^2$ on pienin mahdollinen ja tämän seurauksena matriisien L, R, F ja G täytyy olla nollamatriiseja, sillä muuten C olisi parempi approksimaatio. Näin ollen matriisin A hajotelma yksinkertaistuu muotoon

$$A = U \begin{bmatrix} E & O \\ O & H \end{bmatrix} V^T.$$

Koska $\|A-X\|_F^2$ on pienin mahdollinen, niin lauseen 3.3 nojalla

$$\begin{aligned}
\|A-X\|_F^2 &= \left\| U \begin{bmatrix} E & O \\ O & H \end{bmatrix} V^T - U \begin{bmatrix} D & O \\ O & O \end{bmatrix} V^T \right\|_F^2 \\
&= \left\| \begin{bmatrix} E & O \\ O & H \end{bmatrix} - \begin{bmatrix} D & O \\ O & O \end{bmatrix} \right\|_F^2
\end{aligned}$$

on pienin mahdollinen, kun $E = D$. Tämä johtaa siihen, että matriisin H täytyy koostua matriisin A singulaariarvoista $\sigma_{k+1}, \ldots, \sigma_r$, koska D koostui singulaariarvoista $\sigma_1, \ldots, \sigma_k$, joten normin neliön minimoimiseksi jäljelle jää ainoastaan nuo loput singulaariarvot. Siispä

$$\|A - X\|_F = \|H\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}.$$

Löydettiin siis eräs ratkaisu $X = A'$, kuten alun perin haluttiin osoittaa. □

Nyt siis tekemällä matriisille singulaariarvohajotelman ja muuttamalla käytettävien singulaariarvojen määrää matriisin uudelleenkokoamisessa saadaan eriasteisia approksimaatioita alkuperäiselle matriisille. Tulos kuulostaa varsin hyödylliseltä, ja sitä onkin mahdollista hyödyntää erilaisissa käytännön sovelluksissa.

Lause 3.6 osoitettiin vain Frobenius-normin tapauksessa. Sama lause kuitenkin pätee myös monille muille matriisinormeille, ja hyvää luettavaa matriisinormeista ja todistuksien perusideoista löytyy kirjasta [**15**, sivut 71–73]. Seuraava huomautus sanoo, että Eckartin ja Youngin lauseen voi todistaa mille tahansa matriisinormille, joille pätee lausetta 3.3 vastaava tulos.

Huomautus 3.7. Eckartin ja Youngin lause on mahdollista osoittaa mille tahansa *unitaarisesti invariantille* normille, joille muiden normin ehtojen lisäksi pätee $\|UAV\| = \|A\|$, kun A on $m \times n$-matriisi, U $m \times m$-matriisi ja V $n \times n$-matriisi, ja U ja V ovat unitaarisia. Reaaliavaruudessa pysyttäessä puhutaan *ortogonaalisesti* invarianttiudesta. Lauseen ensimmäisenä yleisti Leon Mirsky. Asiasta enemmän kiinnostuneille suositeltavaa on tutustua julkaisuun [**13**].

Parasta alempiasteista approksimaatiomatriisia voidaan hyödyntää usein eri tavoin. Eräs mielenkiintoinen nykyaikainen sovellus löytyy tekoälypuolelta, jossa alempiasteisten matriisien käyttö on osoittautunut erittäin hyödylliseksi laskentatehokkuuden kannalta.

Huomautus 3.8. *Low-Rank Adaptation* (LoRA) mahdollistaa suurten neuroverkkojen, kuten kielimallien tehokkaan mukauttamisen eri tehtäviin vähentämällä parametrien määrää alempiasteisen approksimaation avulla. Koska LoRA ei alempiasteisesta matriisiapproksimaatiosta huolimatta perustu singulaariarvohajotelmaan, jätetään tarkempi tutkiminen lukijalle. LoRA on esitelty julkaisussa [**9**].

LoRA ja astehajotelma perustuvat samaan perusideaan. Astehajotelman mukaan jokainen astetta r oleva $m \times n$-matriisi A voitiin esittää kahden täyttä astetta olevan matriisin tulona. Jos $k < r$, niin toinen tapa esittää lemman 3.5 minimointiongelma on jakaa matriisi S kahdeksi alempiasteiseksi matriisiksi

$$\min_{X,Y} \|A - XY\|, \tag{3.1}$$

missä X on $m \times k$-matriisi ja Y on $k \times n$-matriisi, ja $\operatorname{rank}(X) = \operatorname{rank}(Y) = k$. Laskennallisesti tämä minimointiongelma voidaan ratkaista iteroimalla, esimerkiksi käyttäen gradienttimenetelmää tai alternoivan minimoinnin avulla, missä vuorotellen kiinnitetään toinen muuttuja ja minimoidaan toista, kunnes päädytään globaaliin

minimiin. Minimointimenetelmistä on tarkemmin kerrottu ainakin kirjassa [**1**, kohta 3.6.4]. Luvussa 5 jatketaan samasta minimointiongelmasta singulaariarvohajotelman näkökulmasta.

Singulaariarvohajotelmasta puhuttaessa matematiikan kannalta yksi tyypillisimmistä esimerkeistä parhaan approksimaation hyödyntämisessä on yleistetyn käänteismatriisin eli pseudoinverssin etsiminen, mikä on yksi tapa löytää pienimmän neliösumman ratkaisu jollekin yhtälöryhmälle $Ax = b$. Tästä kerrotaan enemmän esimerkiksi kirjassa [**14**, sivut 625–628]. Keskitytään tässä tutkielmassa hieman toisenlaisiin sovelluksiin, ja tutkitaan singulaariarvohajotelman hyödyntämistä seuraavaksi kuvanpakkauksen näkökulmasta.

LUKU 4

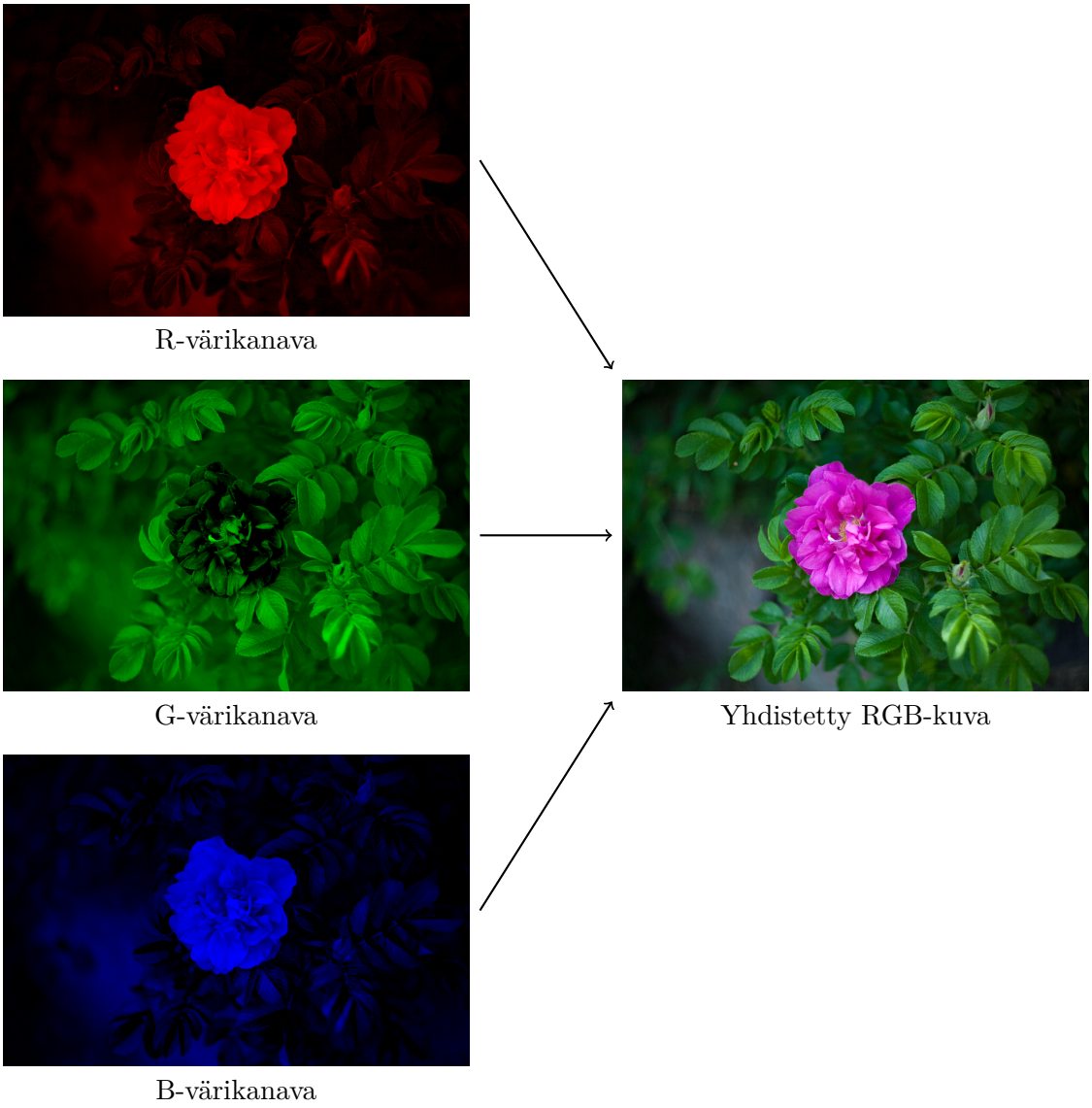
Kuvanpakkaus

Yksi kuva kertoo enemmän kuin tuhat sanaa. Nykyisin valokuvia säilytetään suurimmaksi osaksi digitaalisessa muodossa, ja pöytälaatikon sijasta kuvat vievät tilaa käyttäjien omilta kiintolevyiltä, tai nykyisin enemmissä määrin suurten yritysten palvelimilta. Vaikka kiintolevytila ei välttämättä täyttyisi pelkästään valokuvista, tutkitaan tässä luvussa mahdollisuutta valokuvien pakkaamista pienempään kokoon singulaariarvohajotelman avulla. Samasta aiheesta on kerrottu myös kirjoissa [**12**] sekä [**14**], ja tarkemmin digitaalisten valokuvien toiminnasta ja käsittelystä löytyy teoksesta [**6**].

Digitaaliset valokuvat koostuvat useista kuvapisteistä eli *pikseleistä*. Mitä enemmän pikseleitä kuvassa on, sitä enemmän informaatiota se sisältää. Pikseleistä koostuvaa kuvaa voidaan käsitellä matriiseina. Harmaasävykuvat tallennetaan tietokoneella tyypillisesti 8-bittisinä, eli jokaiselle pikselille on varattu 8 bittiä muistia. Käytännössä siis jokainen pikseli voi esittää $2^8 = 256$ eri arvoa. Nämä arvot vaihtelevat välillä $[0, 255]$, missä arvo 0 tarkoittaa täysin mustaa pikseliä ja 255 valkoista, ja näiden välillä olevat arvot kuvaavat eri harmaan sävyjä mustasta valkoiseen. Harmaasävykuvassa siis kukin matriisin alkio vastaa yhden pikselin harmaasävyarvoa.

Harmaasävykuvien käsittely olisi turhan yksinkertaista, joten keskitytään värikuviin. Yleisin tapa kuvata värejä tietokoneen näytöllä ja värikuvissa on käyttää RGB-värimallia, jossa värit saadaan kolmen eri *värikanavan* avulla: R (red, punainen), G (green, vihreä) ja B (blue, sininen). Kuten harmaasävykuvien tapauksessa, nämä kolme värikanavaa saavat myös arvoja väliltä $[0, 255]$. Esimerkiksi punaisessa värikanavassa 0 tarkoittaa täysin mustaa ja 255 täysin punaista. Värikanavat toimivat omina matriiseinaan, ja lopullinen pikseli muodostuu kolmen värikanavan arvojen perusteella. Eri värivaihtoehtoja tässä värimallissa siis on $256^3 = 16777216$ eli noin 16,8 miljoonaa.

Tutkitaan itse otettua värivalokuvaa hansaruususta, joka näkyy kuvassa 4.1 oikealla. Samassa kuvassa näkyy eriteltynä valokuvan kolme eri värikanavaa, joista lopullinen valokuva muodostuu. Huomataan esimerkiksi, että keskellä kuvaa kukassa on paljon punaista ja sinistä, kun taas reunoilla lehdissä tarvitaan enimmäkseen vain vihreää väriä. Oikeiden värisävyjen luomiseksi kaikkia värikanavia hyödynnetään, ja vaikka lehdet ovatkin silminnähden vihreitä, tarvitsevat niitä esittävät pikselit myös R- sekä B-värikanavaa. Kyseisen värivalokuvan koko pikseleinä ilmoitettuna on 800×1200, missä 800 pikseliä on kuvan korkeus ja 1200 pikseliä leveys. Yleensä valokuvien koko ilmoitetaan muodossa leveys $\times$ korkeus, mutta ilmaistaan nyt matriisien mukaisesti toisin päin. Kuva koostuu siis kolmesta matriisista R, G ja B, jotka kaikki ovat 800×1200-matriiseja. Kullekin matriisille on olemassa singulaariarvohajotelma eli SVD lauseen 2.1 nojalla. Eckartin ja Youngin lauseen 3.6 perusteella saadaan paras approksimaatio



KUVA 4.1. RGB-värikuvan muodostuminen kolmesta värikanavasta.

alempiasteisella matriisilla hyödyntäen singulaariarvohajotelmaa. Hyödynnetään tätä nyt kaikkiin kolmeen värikanavamatriisiin. Jokaisen matriisin R, G, B aste on 800, ja lukija voi halutessaan tarkistaa tämän esimerkiksi Pythonilla. Singulaariarvoista suurimmat ovat tärkeimpiä matriisien uudelleenluonnille ja kuvan uudelleenkokoamiselle, kun taas pienillä arvoilla on hyvin vähän vaikutusta lopulliseen kuvaan. Seuraavaksi esiteltävä Python-funktio avaa alkuperäisen kuvatiedoston Pillow-kirjaston avulla ja muuttaa sen matriisimuotoon NumPy-kirjastosta löytyvällä funktiolla, laskee singulaariarvohajotelman eri värikanaville ja kokoaa kuvan uudelleen käyttäjän määräämällä singulaariarvojen määrällä. Python-ohjelma kokonaisuudessaan tarkkoine kommentteineen löytyy liitteestä A.

```
import numpy as np
import PIL.Image as img
```

```
def svd_kuvanpakkaus(kuvatiedosto, k):
    kuva = img.open(kuvatiedosto)
    kuva_taulukkona = np.array(kuva)
    R = kuva_taulukkona[:, :, 0]
    G = kuva_taulukkona[:, :, 1]
    B = kuva_taulukkona[:, :, 2]
    U_R, Sigma_R, VT_R = np.linalg.svd(R, full_matrices=False)
    U_G, Sigma_G, VT_G = np.linalg.svd(G, full_matrices=False)
    U_B, Sigma_B, VT_B = np.linalg.svd(B, full_matrices=False)
    R_k = np.dot(U_R[:, :k], np.dot(np.diag(Sigma_R[:k]), VT_R[:k, :]))
    G_k = np.dot(U_G[:, :k], np.dot(np.diag(Sigma_G[:k]), VT_G[:k, :]))
    B_k = np.dot(U_B[:, :k], np.dot(np.diag(Sigma_B[:k]), VT_B[:k, :]))
    pakattu_kuva = np.stack([R_k, G_k, B_k], axis=-1)
    pakattu_kuva = np.clip(pakattu_kuva, 0, 255).astype(np.uint8)
    return pakattu_kuva
```

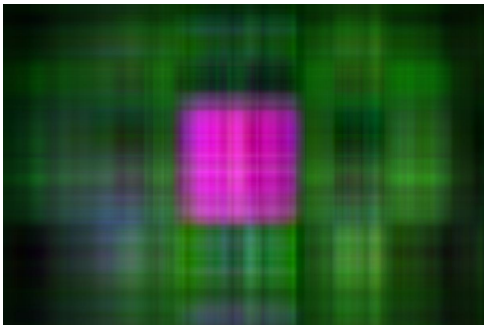
Käytettävä singulaariarvojen määrä eli k voidaan valita joko tutkimalla singulaariarvojen suuruuksia, tai ihan vain kokeilemalla. Esimerkiksi 250 suurimman singulaariarvon approksimaatio saadaan seuraavien ohjelmarivien avulla, kun kuvan tiedostonimi on `kuva.jpg`.

```
kuvatiedosto = 'kuva.jpg'
k = 250
pakattu_kuva = svd_kuvanpakkaus(kuvatiedosto, k)
tallennettava_kuva = img.fromarray(pakattu_kuva).save(f'svd_k{k}.jpg')
```

Kuvassa 4.2 on esimerkkinä käytetyn hansaruusu-kuvan approksimaatioita eri singulaariarvojen määrillä. Visuaalisesti tarkastelemalla huomataan ensimmäisten approksimaatioiden olevan kovin epäselviä, ja informaation määrä kuvissa kasvaa singulaariarvojen määrää kasvattamalla. Jo $k = 100$ antaa kohtuullisen näköisen rekonstruktion alkuperäisestä kuvasta, ainakin jos tarpeeksi kaukaa katsoo. Tarkkasilmäiset näkevät selvän eron vielä $k = 250$ ja alkuperäisen kuvan välilläkin.

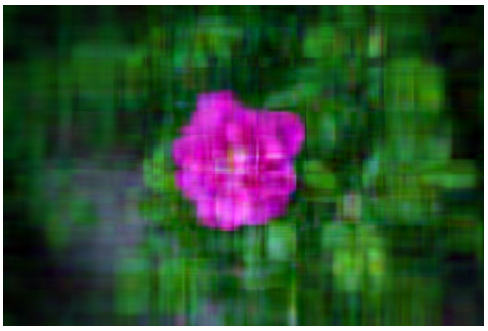
Pohditaan digikuvan viemää tallennustilan määrää matriisien alkioiden eli lukujen näkökulmasta. Kutakin pikseliä vastaa kolme eri matriisin lukua. Esimerkkikuvan koko oli 800×1200, joten kuvan tallentamiseen tarvitaan tieto $3 \cdot 800 \cdot 1200 = 2880000$ eri luvusta. Verrataan tätä 100 suurimman singulaariarvon antamaan approksimaatioon. Kuvanpakkauksen näkökulmasta järkevintä on tallentaa kunkin värikanavamatriisin singulaariarvohajotelmasta itse singulaariarvot sekä vasemmat ja oikeat singulaarivektorit aina indeksiin 100 asti, jolloin tarvittaessa eli haluttaessa tarkastella kuvaa voidaan se luoda näiden tallennettujen osien avulla laskemalla matriisitulot. Tällöin tallennettavia lukuja on vain $3(800 \cdot 100 + 100 + 100 \cdot 1200) = 600300$, eli noin viidesosa alkuperäisestä. Tämä tietysti vaikuttaa kuvanlaatuun, ja singulaariarvojen määrän valinta onkin tasapainoilua tilansäästön ja kuvanlaadun välillä. Seuraavassa luvussa tasapainotellaan suositusten tarkkuuden ja mallin yleistettävyyden välillä, kun perehdytään suosittelujärjestelmän rakentamiseen singulaariarvohajotelman avulla.

Alkuperäinen, $k = r = 800$

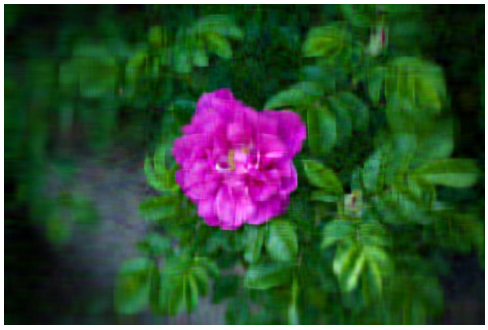


$k = 2$

$k = 5$



$k = 10$



$k = 25$

$k = 50$

$k = 100$

$k = 250$

KUVA 4.2. Singulaariarvojen määrän vaikutus esimerkkivalokuvan laatuun ja yksityiskohtiin.

LUKU 5

Suosittelujärjestelmä

Ihmisistä ja asioista kerätään tänä päivänä valtavasti informaatiota. Digitaalisissa palveluissa pystytään keräämään monipuolista dataa käyttäjistä esimerkiksi ostohistorian, mieltymysten tai vuorovaikuttamisen perusteella. Kaikki tämä data antaa mahdollisuuden parantaa käyttäjien kokemusta ja yrityksen silmistä katsottaessa liiketoiminnan tehokkuutta. Suosittelujärjestelmät ovat yksi erinomainen tapa hyödyntää suuria datamääriä tarjotakseen käyttäjille esimerkiksi räätälöityjä suosituksia yrityksen tarjonnasta.

Suosittelujärjestelmissä dataa voidaan käsitellä matriiseina, joissa käyttäjät ja tuotteet ovat matriisin riveillä ja sarakkeilla. Matriisin alkioihin tallennetaan käyttäjien vuorovaikutukset tuotteiden kanssa, esimerkiksi arvioinnit. Tässä luvussa perehdytään tarkemmin singulaariarvohajotelman hyödyntämiseen suosittelujärjestelmän luomiseksi, ja päälähteinä on käytetty lähteitä [**1**], [**3**], [**8**] sekä [**16**].

Datamatriisia voi olla kätevä approksimoida jollain alempiasteisella matriisilla, jotta valtava tietomäärä saataisiin tiiviimpään muotoon ja helpommin analysoitavaksi. Monesti data sisältää paljon analyysimenetelmien kannalta turhaa tai ei-niin olennaista tietoa, jota kutsutaan joskus *kohinaksi*. Dataa analysoitaessa alempiasteisella matriisilla halutaan kuitenkin pysytellä tarpeeksi lähellä alkuperäistä matriisia, jotta kyetään tunnistamaan ja erottamaan pääpiirteet datasta mahdollisia suositteluja varten. Olkoon R alkuperäinen $m \times n$-datamatriisi astetta r, jonka rivit ovat kunkin käyttäjän arvosteluja kunkin sarakkeen määräämistä asioista, esimerkiksi elokuvista. Olkoon S $m \times n$-matriisi korkeintaan astetta $k < r$. Tämä voidaan esittää pienimmän neliösumman avulla minimointiongelmana:

$$\min_S \|R - S\|_F^2, \quad \text{kun} \quad \operatorname{rank}(S) \leq k.$$

Aste-rajoite voidaan esittää astehajotelman ja kohdan (3.1) tapaan jakamalla matriisi S kahden matriisin tuloksi $S = XY$, missä X on $m \times k$-matriisi ja Y on $k \times n$-matriisi. Tällöin minimointiongelma on muotoa:

$$\min_{X,Y} \|R - XY\|_F^2. \tag{5.1}$$

Matriisit X ja Y eivät luonnollisestikaan ole yksikäsitteisiä. Kun m riviä kuvasti datamatriisissa eri käyttäjiä, voidaan matriisin X ajatella kuvaavan käyttäjäpiirteitä eli esimerkiksi mieltymyksiä. Matriisin X sarakkeita voidaan sanoa olevan käyttäjiin liittyviä piilomuuttujia, eli jokainen sarake vastaa yhtä ominaisuutta. Näitä piilomuuttujia tai -ominaisuuksia ei yleensä suoraan voida havaita alkuperäisestä datasta, vaan ne opitaan datan käsittelyn kautta eli tässä tapauksessa jakamalla matriisi kahteen

osaan. Vastaavasti matriisin Y voidaan ajatella sisältävän elokuviin liittyviä piilomuuttujia riveillä, ja ne voivat kuvata esimerkiksi sarakkeiden elokuvien tyylilajeja tai muita elokuviin liittyviä piirteitä.

Luvussa 3 osoitettiin parhaan alempiasteisen matriisiapproksimaation löytyvän singulaariarvohajotelman avulla, ja sovelletaan sitä seuraavaksi. Olkoon matriisilla R kompakti SVD muotoa $R = U_r \Sigma_r V_r^T$, missä U_r ja V_r^T ovat ortonormaaleja ja Σ_r sisältää matriisin R singulaariarvot päädiagonaalilla laskevassa järjestyksessä. Koska Frobenius-normi oli lauseen 3.3 nojalla ortogonaalisesti invariantti, niin minimointiongelma (5.1) saadaan muotoon

$$\|R - XY\|_F^2 = \left\|U_r \Sigma_r V_r^T - XY\right\|_F^2 = \left\|\Sigma_r - U_r^T XY V_r\right\|_F^2 .$$

Halutaan siis löytää matriisi $U_r^T XY V_r$ korkeintaan astetta k, joka on lähimpänä singulaariarvomatriisia Σ_r. Helposti nähdään, että Eckartin ja Youngin lauseesta 3.6 saadaan paras approksimaatio Σ_k, mikä siis sisältää alkuperäisen singulaariarvomatriisin Σ_r k suurinta singulaariarvoa. Haluttu approksimaatio saadaan, kun valitaan $U_r^T XY V_r = \Sigma_k$, tai mieluummin $XY = U_r \Sigma_k V_r^T$. Määritellään nyt

$$U_k = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{bmatrix}, \qquad V_k = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_k \\ | & & | \end{bmatrix},$$

ja olkoot

$$X = U_k \sqrt{\Sigma_k} \qquad \text{ja} \qquad Y = \sqrt{\Sigma_k} V_k^T .$$

Näin ollaan saatu minimointiongelmalle ratkaisu singulaariarvohajotelmaan perustuen, ja

$$\|R - XY\|_F^2 = \left\|R - U_k \sqrt{\Sigma_k} \sqrt{\Sigma_k} V_k^T\right\|_F^2 = \left\|R - U_k \Sigma_k V_k^T\right\|_F^2 . \tag{5.2}$$

Paras astetta k oleva approksimaatio datamatriisille R saadaan siis singulaariarvohajotelman avulla. Otetaan seuraavaksi esimerkki pienelle datamatriisille. Olkoon R 4×5-matriisi, jonka rivit koostuvat neljän eri käyttäjän antamista arvosteluista elokuville, joista kolme ensimmäistä saraketta kuvaavat eri komediaelokuvia, ja loput kaksi saraketta kauhuelokuvia. Sovitaan, että arvosteluasteikko on $[1, 5]$. Koska kaikki käyttäjät eivät katso ja arvostele välttämättä kaikkia elokuvia, on tässäkin esimerkkimatriisissa tyhjiä alkioita, joita on merkitty kysymysmerkillä. Olkoon elokuva-arvostelumatriisi nyt

$$\begin{array}{c} \qquad\quad \overbrace{\text{komedia}}^{} \quad \overbrace{\text{kauhu}}^{} \\ \begin{array}{l} \text{Aapeli} \\ \text{Bertta} \\ \text{Cecilia} \\ \text{Daniel} \end{array} \begin{bmatrix} 4 & 5 & 4 & 2 & 1 \\ 1 & ? & 2 & 5 & 4 \\ ? & 4 & 5 & 2 & 1 \\ 2 & 1 & 1 & 4 & ? \end{bmatrix} \end{array} := R.$$

Esimerkiksi käyttäjä Bertta ei ole arvostellut toista komediaelokuvaa. Lisäksi huomataan, että Aapeli ja Cecilia näyttävät pitävän enemmän komediaelokuvista, kun taas Bertta ja Daniel ovat pitäneet enemmän kauhuelokuvista. Tyhjät alkiot tulee täyttää jollain tavalla, jotta singulaariarvohajotelma voidaan tehdä kyseiselle matriisille.

Kokeillaan ensin asettaa tuntemattomat alkiot nolliksi, ja käytetään merkintää R_0. Täydennetty matriisi on siis

$$R_0 = \begin{bmatrix} 4 & 5 & 4 & 2 & 1 \\ 1 & 0 & 2 & 5 & 4 \\ 0 & 4 & 5 & 2 & 1 \\ 2 & 1 & 1 & 4 & 0 \end{bmatrix}.$$

Nyt tälle täydennetylle matriisille R_0 pystytään tekemään singulaariarvohajotelma. Hyödynnetään hajotelman tekemiseen Pythonin NumPy-kirjastoa, jolla saadaan kompakti SVD -muoto kahden desimaalin tarkkuudella ilmaistuna

$$R_0 = \begin{bmatrix} 0{,}64 & 0{,}42 & 0{,}45 & -0{,}46 \\ 0{,}44 & -0{,}79 & -0{,}24 & -0{,}35 \\ 0{,}53 & 0{,}33 & -0{,}67 & 0{,}40 \\ 0{,}33 & -0{,}30 & 0{,}54 & 0{,}71 \end{bmatrix} \begin{bmatrix} 11{,}42 & 0 & 0 & 0 \\ 0 & 5{,}57 & 0 & 0 \\ 0 & 0 & 3{,}30 & 0 \\ 0 & 0 & 0 & 1{,}92 \end{bmatrix}$$
$$\begin{bmatrix} 0{,}32 & 0{,}50 & 0{,}56 & 0{,}52 & 0{,}26 \\ 0{,}06 & 0{,}56 & 0{,}26 & -0{,}65 & -0{,}43 \\ 0{,}80 & 0{,}03 & -0{,}45 & 0{,}15 & -0{,}36 \\ -0{,}39 & 0{,}01 & 0{,}09 & 0{,}51 & -0{,}76 \end{bmatrix}.$$

Huomataan, että kaksi ensimmäistä singulaariarvoa ovat suurempia kuin loput kaksi, joten niillä on enemmän painoarvoa alkuperäisen matriisin uudelleenluomisessa. Ensimmäisen singulaariarvon voidaan ajatella kuvaavan komediaelokuvien painoarvoa, sillä se on suurempi ja komediaelokuvia oli matriisissa eniten. Toisaalta asetettaessa pienimmät kaksi singulaariarvoa nolliksi voidaan matriisit jakaa lausekkeen (5.2) mukaisesti käyttäjäpiirteiden matriisiin X_0 ja elokuvapiirteiden matriisiin Y_0, jolloin

$$X_0 = \begin{bmatrix} 0{,}64 & 0{,}42 & 0{,}45 & -0{,}46 \\ 0{,}44 & -0{,}79 & -0{,}24 & -0{,}35 \\ 0{,}53 & 0{,}33 & -0{,}67 & 0{,}40 \\ 0{,}33 & -0{,}30 & 0{,}54 & 0{,}71 \end{bmatrix} \begin{bmatrix} \sqrt{11{,}42} & 0 & 0 & 0 \\ 0 & \sqrt{5{,}57} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 2{,}16 & 1{,}00 \\ 1{,}50 & -1{,}86 \\ 1{,}80 & 0{,}78 \\ 1{,}11 & -0{,}71 \end{bmatrix},$$
$$Y_0 = \begin{bmatrix} \sqrt{11{,}42} & 0 & 0 & 0 \\ 0 & \sqrt{5{,}57} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0{,}32 & 0{,}50 & 0{,}56 & 0{,}52 & 0{,}26 \\ 0{,}06 & 0{,}56 & 0{,}26 & -0{,}65 & -0{,}43 \\ 0{,}80 & 0{,}03 & -0{,}45 & 0{,}15 & -0{,}36 \\ -0{,}39 & 0{,}01 & 0{,}09 & 0{,}51 & -0{,}76 \end{bmatrix}$$
$$= \begin{bmatrix} 1{,}08 & 1{,}68 & 1{,}91 & 1{,}74 & 0{,}87 \\ 0{,}13 & 1{,}33 & 0{,}62 & -1{,}54 & -1{,}02 \end{bmatrix}.$$

Muodostetaan nyt approksimaatio matriisista R_0 käyttäen kahta suurinta singulaariarvoa eli laskemalla matriisitulo X_0Y_0, jolloin kahden desimaalin tarkkuudella saadaan

$$R_0' = \begin{bmatrix} 2{,}48 & 4{,}96 & 4{,}75 & 2{,}23 & 0{,}87 \\ 1{,}38 & 0{,}03 & 1{,}70 & 5{,}47 & 3{,}20 \\ 2{,}06 & 4{,}06 & 3{,}93 & 1{,}94 & 0{,}78 \\ 1{,}12 & 0{,}93 & 1{,}68 & 3{,}02 & 1{,}69 \end{bmatrix}.$$

Selvästi nähdään, että tuntemattomien alkioiden valitseminen nolliksi vaikuttaa singulaariarvohajotelman toimimiseen. Olisi esimerkiksi voinut kuvitella, että koska Cecilia oli arvostellut kaksi jälkimmäistä komediaelokuvaa arvoilla 4 ja 5, tulisi myös ensimmäiselle komediaelokuvalle muodostumaan teoreettisesti ajateltuna korkea arvio, tai ei ainakaan noin pieni 2,06. Toisaalta Daniel oli antanut ensimmäiselle kauhuelokuvalle arvosanan 4, mutta tässä approksimaatiossa kyseinen arvo on pudonnut melkein yhden kokonaisluvun verran, ja toiselle kauhuelokuvalle on tullut vielä tätäkin pienempi arvio 1,69. Nollilla täydentäminen ei siis ainakaan suoraan toimi kovin hyvin, sillä nollan ollessa arvosteluasteikon alapuolella lähtevät muiden matriisin alkioiden arvot myös herkästi laskemaan muodostettaessa alempiasteista approksimaatiomatriisia.

Kokeillaan toista tapaa täydentää matriisi R nolla-alkioiden sijasta kunkin rivin tyhjät arviot kyseisen käyttäjän henkilökohtaisten arvosteluiden keskiarvoilla. Tällöin saadaan

$$R_{\overline{x}} = \begin{bmatrix} 4 & 5 & 4 & 2 & 1 \\ 1 & 3 & 2 & 5 & 4 \\ 3 & 4 & 5 & 2 & 1 \\ 2 & 1 & 1 & 4 & 2 \end{bmatrix},$$

ja käyttäjien keskiarvoilla täydennetyn matriisin $R_{\overline{x}}$ singulaariarvohajotelmaksi kahden desimaalin tarkkuudella

$$R_{\overline{x}} = \begin{bmatrix} -0{,}58 & -0{,}44 & 0{,}21 & -0{,}65 \\ -0{,}51 & 0{,}65 & -0{,}54 & -0{,}17 \\ -0{,}55 & -0{,}41 & -0{,}21 & 0{,}70 \\ -0{,}33 & 0{,}46 & 0{,}79 & 0{,}24 \end{bmatrix} \begin{bmatrix} 12{,}92 & 0 & 0 & 0 \\ 0 & 5{,}20 & 0 & 0 \\ 0 & 0 & 1{,}65 & 0 \\ 0 & 0 & 0 & 1{,}12 \end{bmatrix}$$
$$\begin{bmatrix} -0{,}39 & -0{,}54 & -0{,}49 & -0{,}47 & -0{,}30 \\ -0{,}28 & -0{,}28 & -0{,}40 & 0{,}65 & 0{,}51 \\ 0{,}77 & -0{,}36 & -0{,}28 & 0{,}28 & -0{,}35 \\ -0{,}16 & -0{,}64 & 0{,}71 & 0{,}20 & -0{,}13 \end{bmatrix}.$$

Kokeillaan uudelleen approksimoida alkuperäistä matriisia valitsemalla kaksi suurinta singulaariarvoa, jolloin saadaan kahdella desimaalilla ilmaistuna

$$R_{\overline{x}}' = \begin{bmatrix} 3{,}61 & 4{,}66 & 4{,}62 & 2{,}04 & 1{,}03 \\ 1{,}66 & 2{,}56 & 1{,}88 & 5{,}29 & 3{,}66 \\ 3{,}39 & 4{,}38 & 4{,}34 & 1{,}94 & 0{,}98 \\ 1{,}05 & 1{,}64 & 1{,}17 & 3{,}59 & 2{,}49 \end{bmatrix}.$$

Saatu approksimaatio vaikuttaa uskottavammalta, sillä esimerkiksi Cecilialle saatu arvio ensimmäiselle komediaelokuvalle on paremmin linjassa kahden muun jo arvioidun komediaelokuvan kanssa. Toisaalta Danielille saatu arvio toisen kauhuelokuvan

arvioksi on ymmärrettävästi hieman alhaisempi kuin ensimmäisen kauhuelokuvan, sillä kaikki muutkin ovat arvioineet kyseistä elokuvaa huonommaksi riippumatta siitä, onko kauhuelokuvien ystävä vai ei.

Ideana ei ollut vaikuttaa käyttäjien jo arvosteltuihin elokuviin, vaan löytää arviot tyhjille alkioille. Nyt matriisin $R'_{\overline{x}}$ perusteella voidaan muodostaa alkuperäisen matriisin täydentävä kokonaisuus

$$\hat{R}_{\overline{x}} = \begin{bmatrix} 4 & 5 & 4 & 2 & 1 \\ 1 & 2{,}56 & 2 & 5 & 4 \\ 3{,}39 & 4 & 5 & 2 & 1 \\ 2 & 1 & 1 & 4 & 2{,}49 \end{bmatrix}.$$

Rivikeskiarvojen käyttäminen ei kuitenkaan ole täydellinen ratkaisu, sillä käyttäjät voivat arvostella elokuvia hyvinkin erilaisin menetelmin. Toisaalta jotkut elokuvat saattavat saada lähes poikkeuksetta muita elokuvia korkeampia arvosteluja, mutta rivikeskiarvoa käyttämällä tätä ei oteta huomioon. Alkuarvauksia voidaan tehdä monin eri keinoin, ja useista eri tavoista on kerrottu esimerkiksi julkaisussa [**5**]. Alkuarvauksista johtuvaa vinoumaa voidaan koittaa vähentää myös lisäämällä alkuperäiseen minimointiongelmaan regularisointitermit matriiseille X ja Y. Minimointiongelma (5.1) saa tällöin muodon

$$\min_{X,Y} \|R - XY\|_F^2 + \gamma \|X\|_F^2 + \gamma \|Y\|_F^2 ,$$

missä $\gamma \geq 0$ on regularisointikerroin. Jos $\gamma = 0$, palautuu ongelma aiempaan muotoon (5.1). Osoitetaan seuraavaksi, että kyseisen minimointiongelman eräs ratkaisu saadaan myös, ihme kyllä, singulaariarvohajotelman avulla.

Lause 5.1. *Olkoon R $m \times n$-matriisi astetta r, jolla on kompakti SVD muotoa $R = U_r \Sigma_r V_r^T$, missä U ja V ovat ortogonaaliset matriisit ja Σ on diagonaalimatriisi sisältäen matriisin R singulaariarvot. Olkoon XY korkeintaan $k < r$ astetta oleva matriisi, missä X on $m \times k$-matriisi ja Y on $k \times n$-matriisi, ja olkoon $\gamma \geq 0$ regularisointikerroin, jolloin minimointiongelman*

$$\min_{X,Y} \|R - XY\|_F^2 + \gamma \|X\|_F^2 + \gamma \|Y\|_F^2$$

globaali minimi saadaan ratkaisusta

$$X = U_k \sqrt{\tilde{\Sigma}_k}, \quad Y = \sqrt{\tilde{\Sigma}_k} V_k^T,$$

missä

$$U_k = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{bmatrix}, \qquad V_k = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_k \\ | & & | \end{bmatrix},$$

ja

$$\tilde{\Sigma} = \begin{bmatrix} \max(\sigma_1 - \gamma, 0) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \max(\sigma_k - \gamma, 0) \end{bmatrix}.$$

TODISTUS. Tämä todistus on mukailtu julkaisusta [**16**, ratkaisu A.1.1]. Olkoon nyt

$$X = \begin{bmatrix} — & x_1 & — \\ & \vdots & \\ — & x_m & — \end{bmatrix} \quad \text{ja} \quad Y = \begin{bmatrix} | & & | \\ y_1 & \cdots & y_n \\ | & & | \end{bmatrix},$$

missä $x_p, y_q \in \mathbb{R}^k$ kaikilla $1 \leq p \leq m$ ja $1 \leq q \leq n$. Minimointiongelman ratkaisun eli minimin löytämiseksi on etsittävä kriittiset pisteet eli ääriarvokohdat. Muodostetaan minimointiongelmasta nyt funktio $f\colon \mathbb{R}^{k(m+n)} \to \mathbb{R}$, missä

$$f(x_1, \ldots, x_m, y_1, \ldots, y_n) = \sum_{i=1}^{m} \sum_{j=1}^{n} (R_{ij} - x_i y_j)^2 + \gamma \sum_{i=1}^{n} \|x_i\|_2^2 + \gamma \sum_{j=1}^{n} \|y_j\|_2^2 .$$

Merkitään $s := (x_1, \ldots, x_m, y_1, \ldots, y_n)$. Funktio f on selvästi jatkuva, sillä $f(s) \geq 0$ kaikilla s ja $\lim\limits_{\|s\| \to \infty} f(s) = \infty$, joten funktio saavuttaa pienimmän arvonsa.

Etsitään kriittiset pisteet differentiaalilaskennan ja tarkemmin osittaisderivaattojen avulla. Olkoot $1 \leq p \leq m$ ja $1 \leq t \leq k$, jolloin

$$\begin{aligned} \frac{\partial f}{\partial X_{pt}} &= -2 \sum_{i=1}^{m} \sum_{j=1}^{n} (R_{ij} - x_i y_j)\, \delta_{ip} Y_{tj} + 2\gamma \sum_{i=1}^{m} \sum_{u=1}^{k} \delta_{ip} \delta_{ut} X_{iu} \\ &= -2 \sum_{j=1}^{n} (R_{pj} - x_p y_j) Y_{tj} + 2\gamma X_{pt} \\ &= -2 \left((R - XY) Y^T \right)_{pt} + 2\gamma X_{pt}. \end{aligned}$$

Erityisesti

$$\begin{aligned} & \frac{\partial f}{\partial X_{pt}} = 0 \quad \text{kaikilla} \quad 1 \leq p \leq m, 1 \leq t \leq k \\ \iff \quad & -(R - XY) Y^T + \gamma X = O, \end{aligned}$$

missä O on nollamatriisi. Vastaavalla tavalla saadaan laskettua termit $\partial f / \partial Y_{qt}$, kun $1 \leq q \leq n$, ja

$$\begin{aligned} \frac{\partial f}{\partial Y_{qt}} &= -2 \sum_{i=1}^{m} \sum_{j=1}^{n} (R_{ij} - x_i y_j)\, \delta_{jq} X_{it} + 2\gamma \sum_{j=1}^{n} \sum_{u=1}^{k} \delta_{jq} \delta_{ut} Y_{uj} \\ &= -2 \sum_{i=1}^{m} (R_{iq} - x_i y_q) X_{it} + 2\gamma Y_{tq} \\ &= -2 \left((R - XY)^T X \right)_{qt} + 2\gamma Y_{qt}^T. \end{aligned}$$

Siis erityisesti

$$\begin{aligned} & \frac{\partial f}{\partial Y_{qt}} = 0 \quad \text{kaikilla} \quad 1 \leq q \leq n, 1 \leq t \leq k \\ \iff \quad & -(R - XY)^T X + \gamma Y^T = O. \end{aligned}$$

Näin ollen lausekkeen

$$\|R - XY\|_F^2 + \gamma \|X\|_F^2 + \gamma \|Y\|_F^2$$

kriittiset pisteet eli lokaalit ääriarvokohdat toteuttavat yhtälöt

$$\begin{cases} -(R - XY)Y^T + \gamma X = O \\ -(R - XY)^T X + \gamma Y^T = O. \end{cases}$$

Kertomalla ensimmäistä yhtälöä vasemmalta termillä X^T ja toista yhtälöä termillä Y, ja siirtelemällä termejä saadaan yhtälöpari muotoon

$$\begin{cases} X^T(R - XY)Y^T = \gamma X^T X \\ Y(R - XY)^T X = \gamma Y Y^T. \end{cases}$$

Ottamalla transpoosi esimerkiksi ensimmäisestä yhtälöstä saadaan

$$\begin{aligned} & \big(X^T(R - XY)Y^T\big)^T = \big(\gamma X^T X\big)^T \\ \iff \qquad & Y(R - XY)^T X = \gamma X^T X, \end{aligned}$$

jolloin tämän ja toisen yhtälön perusteella $X^T X = YY^T$. Kirjoitetaan tämän tiedon avulla yhtälöpari nyt blokkimuodossa

$$\begin{aligned} \begin{bmatrix} -\gamma I & R \\ R^T & -\gamma I \end{bmatrix} \begin{bmatrix} X \\ Y^T \end{bmatrix} &= \begin{bmatrix} O & XY \\ (XY)^T & O \end{bmatrix} \begin{bmatrix} X \\ Y^T \end{bmatrix} \\ &= \begin{bmatrix} X\big(YY^T\big) \\ Y^T\big(X^T X\big) \end{bmatrix} \\ &= \begin{bmatrix} X \\ Y^T \end{bmatrix} \big(X^T X\big), \end{aligned}$$

missä yksikkömatriisit I ja nollamatriisit O ovat sopivan kokoisia. Olkoon λ matriisin $X^T X$ ominaisarvo. Blokkimuodosta saadun yhtäsuuruuden nojalla λ on myös matriisin

$$\begin{bmatrix} -\gamma I & R \\ R^T & -\gamma I \end{bmatrix}$$

ominaisarvo. Tämä matriisi on symmetrinen, joten se on diagonalisoituva. Etsitään ominaisarvot karakteristisen polynomin avulla, mistä saadaan

$$\begin{aligned} \det\left(\begin{bmatrix} -\gamma I & R \\ R^T & -\gamma I \end{bmatrix} - \lambda I\right) &= \det\begin{bmatrix} (-\gamma - \lambda)I & R \\ R^T & (-\gamma - \lambda)I \end{bmatrix} \\ &= \det((-\gamma - \lambda)I)\det\big((-\gamma - \lambda)I - R^T((-\gamma - \lambda)I)^{-1}R\big) \\ &= \det((-\gamma - \lambda)I)\det\left((-\gamma - \lambda)I - R^T\left(\frac{1}{-\gamma - \lambda}I\right)R\right) \\ &= \det((-\gamma - \lambda)I)\det\left(\frac{(-\gamma - \lambda)^2 I - R^T R}{-\gamma - \lambda}\right) \\ &= \det\big((-\gamma - \lambda)^2 I - R^T R\big). \end{aligned}$$

Toisella rivillä on käytetty blokkimatriisin determinantin laskusääntöä, joka löytyy kirjasta [**2**, lause 10.10]. Matriisin R singulaariarvot ovat σ_i, kun $1 \leq i \leq k$, joten

matriisin R^TR ominaisarvoja ovat σ_i^2. Näin ollen saadaan

$$\begin{aligned} & (-\gamma-\lambda)^2-\sigma_i^2=0 \\ \iff \quad & (-\gamma-\lambda)^2=\sigma_i^2 \\ \iff \quad & -\gamma-\lambda=\pm\sigma_i \\ \iff \quad & \lambda=-\gamma\pm\sigma_i. \end{aligned}$$

Ominaisarvoja vastaavat ominaisvektorit saadaan helposti matriisin R SVD-muodosta $R=U_r\Sigma_rV_r^T$. Etsitään nyt esimerkiksi ominaisarvoja $\lambda=-\gamma+\sigma_i$ vastaavat ominaisvektorit (w,z):

$$\begin{aligned} & \begin{bmatrix}(-\gamma-(-\gamma+\sigma_i))I & R \\ R^T & (-\gamma-(-\gamma+\sigma_i))I\end{bmatrix}\begin{bmatrix}w\\z\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix} \\ \iff \quad & \begin{bmatrix}-\sigma_iI & R \\ R^T & -\sigma_iI\end{bmatrix}\begin{bmatrix}w\\z\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}. \end{aligned}$$

Lausekkeen (2.3) perusteella saadaan yhtäsuuruudet $Rv_i=\sigma_iu_i$, ja $R^Tu_i=\sigma_iv_i$, joten

$$\begin{cases}-\sigma_iw+Rz=0\\ R^Tw-\sigma_iz=0.\end{cases}$$

Nyt valitsemalla $w=u_i$ ja $z=v_i$ löydetään yhtälöryhmälle ratkaisu äskeisten yhtäsuuruuksien perusteella, sillä

$$\begin{cases}-\sigma_iu_i+Rv_i=0\\ R^Tu_i-\sigma_iv_i=0.\end{cases}$$

Siispä ominaisarvoja $-\gamma+\sigma_i$ vastaavat ominaisvektorit ovat (u_i,v_i). Vastaavalla päättelyllä saadaan pienempiä ominaisarvoja $-\gamma-\sigma_i$ vastaavat ominaisvektorit $(u_i,-v_i)$.

Koska X^TX on positiivisesti semidefiniitti, ovat sen ominaisarvot myös positiivisia. On siis olemassa joukko

$$\Omega=\big\{1\le i\le k\colon \sigma_i-\gamma\ge 0 \text{ on matriisin } X^TX \text{ ominaisarvo}\big\}.$$

Matriisilla X on siis singulaariarvot $\sqrt{\sigma_i-\gamma}$, kun $i\in\Omega$. Koska $X^TX=YY^T$, on matriisilla Y nämä samat singulaariarvot. Nyt

$$\begin{bmatrix}X\\Y^T\end{bmatrix}$$

virittää vektorien (u_i,v_i) määrittämän aliavaruuden. Koska u_i ovat matriisin R vasemmanpuoleiset singulaarivektorit ja v_i oikeanpuoleiset singulaarivektorit, niin ne kelpaavat myös matriisin XY singulaarivektoreiksi, koska ominaisarvoja $-\gamma+\sigma_i$ vastasivat samat ominaisvektorit (u_i,v_i). Näin ollen $XY=\sum_{i\in\Omega}u_i(\sigma_i-\gamma)v_i^T$.

Kuvataan nyt mitä tahansa ääriarvokohtaa merkitsemällä

$$X=U_\Omega\sqrt{\Sigma_\Omega-\gamma I},\quad Y=\sqrt{\Sigma_\Omega-\gamma I}V_\Omega^T,$$

missä alaindeksi Ω tarkoittaa, että alkuperäisistä matriiseista U ja V valitaan ainoastaan sarakkeet $i\in\Omega$, ja diagonaalimatriisista Σ singulaariarvot, joiden indeksit $i\in\Omega$.

Nyt riippumatta regularisointikertoimesta γ, ehdokkaita ääriarvokohdiksi on

$$\sum_{i=0}^{k} \binom{\max\{i\colon \sigma_i \geq \gamma\}}{i},$$

yksi kullekin valitulle Ω. Ääriarvokohtien määrä vähenee lukua γ kasvatettaessa, ja kun $\gamma > \omega_1$, ainoa vaihtoehto on $X = 0, Y = 0$.

Ääriarvokohdissa minimoitava lauseke voi saada hyvinkin erilaisia arvoja. Jos (X, Y) on kuten edellä, niin

$$\|R - XY\|_F^2 + \gamma \|X\|_F^2 + \gamma \|Y\|_F^2 = \sum_{i \notin \Omega} \sigma_i^2 + \sum_{i \in \Omega} \left(\gamma^2 + 2\gamma \left|\sigma_i - \gamma\right|\right).$$

Tästä nähdään, että halutaan valita Ω varustettuna suurimpien singulaariarvojen indekseillä $i = 1, \ldots, \max\{i\colon \sigma_i \geq \gamma\}$. Näin ollaan osoitettu minimointiongelmalle eräs ratkaisu. Itse asiassa muilla osajoukoilla Ω ei päästäisi haluttuun lokaaliin minimiin, joka on myös globaali minimi. Globaalin minimin todistuksen ideana on lisätä pieni muutos $\epsilon > 0$ ääriarvokohdan matriisin XY singulaarivektoreihin u_i ja v_i ja osoittaa, että tämä muutos kasvattaa hieman regularisointitermiä, mutta samalla approksimaation virhe pienenee ja siten alkuperäinen valinta ei ollutkaan lokaali minimi. Tällä tavalla löytyy todellinen lokaali minimi, joka on myös globaali, sillä muiden ehdokkaiden virhe pienenee lisäämällä ϵ singulaarivektoreihin. Tarkemmat perustelut löytyvät julkaisusta [**16**, lause A.1]. □

Tehdään vielä huomautus saadusta ratkaisusta.

Huomautus 5.2. Vaikka lokaali minimi onkin globaali minimi, saatu ratkaisu ei ole yksikäsitteinen. Jos (X, Y) on ääriarvokohta, niin $(XZ, Z^{-1}Y)$ on myös ääriarvokohta, jos

$$\begin{cases} -(R - XY)Y^T(Z^{-1})^T + \gamma XZ = O \\ -(R - XY)^T XZ + \gamma Y^T(Z^{-1})^T = O, \end{cases}$$

mikä on totta aina, kun Z on ortogonaalinen. Ääriarvokohtien joukko on siten invariantti ortogonaalisten muunnosten suhteen.

Minimointiongelman globaali minimi saadaan siis myös muotoiltua singulaariarvohajotelman avulla. Tässä tapauksessa regularisointikerroin γ vähennetään kaikista singulaariarvoista, ja se ikään kuin tasaa kunkin singulaariarvon painoarvoa ennen matriisin uudelleenluomista. Kertoimen valinnalla voi olla paljonkin merkitystä saatavan approksimaation laatuun, ja liian suuri kerroin muuttaa pienimmät singulaariarvot nolliksi. Parhaan regularisointikertoimen löytäminen vaatii approksimoitavan datan ymmärtämistä, ja jos halutaan oikeasti verrata kertoimen vaikutusta muihin matriisintäydentämismenetelmiin, tulisi tyhjien alkioiden arvot tietää entuudestaan. Kuitenkaan elokuva-arvostelujen tapauksessa kaikki käyttäjät eivät arvioi eivätkä katsokaan kaikkia elokuvia. Koneoppimisen yksi tärkeimmistä periaatteista on jakaa data erillisiin opetus- ja testijoukkoihin. Tavoitteena on arvioida käytettävän menetelmän yleistettävyyttä, eli kuinka hyvin menetelmä tai malli pystyy ennustamaan uutta, ennalta näkemätöntä dataa.

Opetusjoukkoa käytetään mallin kouluttamiseen. Se on osa alkuperäistä dataa, josta on poistettu osia testijoukkoa varten. Testijoukko koostuu vain niistä osista, joita ei opetusjoukossa ole. Testijoukko toimii ikään kuin uutena datana, jolla mallin tai menetelmän suorituskykyä voidaan arvioida. Arviointi on mahdollista monin eri keinoin, ja kaksi hyvin soveltuvaa virhemittaria ovat *keskineliövirheen neliöjuuri* (engl. root mean square error, RMSE) sekä *keskimääräinen absoluuttinen virhe* (engl. mean absolute error, MAE).

MÄÄRITELMÄ 5.3. Olkoon R $m \times n$-matriisi, joka koostuu todellisista arvoista, ja olkoon $\hat{R}$ vastaava $m \times n$-ennustematriisi, joka sisältää arvioita valikoiduille alkioille matriisista R. Keskineliövirheen neliöjuuri (RMSE) ja keskimääräinen absoluuttinen virhe (MAE) määritellään seuraavasti:

$$\mathrm{RMSE} = \sqrt{\frac{1}{|S|} \sum_{(i,j) \in S} (\hat{r}_{ij} - r_{ij})^2},$$

$$\mathrm{MAE} = \frac{1}{|S|} \sum_{(i,j) \in S} |\hat{r}_{ij} - r_{ij}|,$$

missä r_{ij} on matriisin R alkio rivillä i ja sarakkeessa j, $\hat{r}_{ij}$ on vastaava ennustematriisin $\hat{R}$ alkio, S on joukko indeksejä (i, j), jotka vastaavat niitä alkioita, joille ennusteet on tehty ja joiden todelliset arvot ovat tiedossa, ja $|S|$ on joukon S alkioiden lukumäärä.

Nyt siis RMSE mittaa keskimääräistä suuruutta opetusjoukon avulla koulutetun mallin tekemän ennusteiden ja testijoukota löytyvien vastaavien todellisten arvojen välillä. Koska kukin virhe korotetaan toiseen potenssiin ennen alkioiden lukumäärällä jakamista ja neliöjuuren ottamista, kasvattavat suuremmat poikkeamat suhteessa enemmän RMSE:tä kuin pienemmät virheet. RMSE on helposti tulkittavissa alkuperäiseen dataan verraten, sillä skaalaus pysyy samana: jos RMSE ≈ 1, niin saadut ennustetut arvot ovat keskimäärin yhden yksikön päässä todellisista arvoista. Siinä missä RMSE kasvaa suurien virheiden vaikutuksesta enemmän, kohtelee MAE kaikkia virheitä samanarvoisesti, antaen yksinkertaisen keskiarvon virheiden suuruudesta.

Tyypillisesti data jaetaan satunnaisesti opetus- ja testijoukkoon. Datasta voitaisiin jakaa esimerkiksi 70–80 % opetusjoukkoon ja 20–30 % testijoukkoon. Näytetään nyt havainnollistava esimerkki aiemmin määritellyn matriisin R avulla. Jaetaan R opetusmatriisiksi L ja testimatriisiksi T siten, että

$$L = \begin{bmatrix} 4 & 5 & 4 & ? & 1 \\ 1 & ? & 2 & ? & 4 \\ ? & ? & 5 & 2 & 1 \\ 2 & 1 & ? & 4 & ? \end{bmatrix} \quad \text{ja} \quad T = \begin{bmatrix} ? & ? & ? & 2 & ? \\ ? & ? & ? & 5 & ? \\ ? & 4 & ? & ? & ? \\ ? & ? & 1 & ? & ? \end{bmatrix}.$$

Täydennetään tuttuun tapaan matriisi L rivikeskiarvoilla, jolloin

$$L_{\overline{x}} = \begin{bmatrix} 4 & 5 & 4 & 3{,}5 & 1 \\ 1 & 2{,}33 & 2 & 2{,}33 & 4 \\ 2{,}67 & 2{,}67 & 5 & 2 & 1 \\ 2 & 1 & 2{,}33 & 4 & 2{,}33 \end{bmatrix}.$$

Tämän matriisin eräs singulaariarvohajotelma kahden desimaalin tarkkuudella ilmaistuna on

$$L_{\overline{x}} = \begin{bmatrix} -0{,}65 & -0{,}42 & 0{,}22 & 0{,}59 \\ -0{,}39 & 0{,}68 & 0{,}59 & -0{,}17 \\ -0{,}51 & -0{,}36 & -0{,}13 & -0{,}77 \\ -0{,}41 & 0{,}47 & -0{,}76 & 0{,}17 \end{bmatrix} \begin{bmatrix} 12{,}51 & 0 & 0 & 0 \\ 0 & 3{,}75 & 0 & 0 \\ 0 & 0 & 2{,}00 & 0 \\ 0 & 0 & 0 & 1{,}90 \end{bmatrix} \begin{bmatrix} -0{,}41 & -0{,}47 & -0{,}55 & -0{,}47 & -0{,}29 \\ -0{,}28 & -0{,}27 & -0{,}28 & 0{,}34 & 0{,}81 \\ -0{,}19 & 0{,}70 & -0{,}17 & -0{,}57 & 0{,}35 \\ 0{,}26 & 0{,}36 & -0{,}75 & 0{,}44 & -0{,}23 \end{bmatrix}.$$

Luodaan taas approksimaatiomatriisi valitsemalla kaksi suurinta singulaariarvoa, mutta kokeillaan regularisointikerrointa $\gamma = 0{,}2$ eli vähennetään molemmista singulaariarvoista 0,2. Tällöin saadaan approksimaatioksi

$$L'_{\overline{x};\gamma=0{,}2} = \begin{bmatrix} 3{,}72 & 4{,}20 & 4{,}82 & 3{,}23 & 1{,}14 \\ 1{,}31 & 1{,}61 & 1{,}96 & 3{,}06 & 3{,}38 \\ 2{,}93 & 3{,}30 & 3{,}79 & 2{,}47 & 0{,}78 \\ 1{,}62 & 1{,}94 & 2{,}31 & 2{,}92 & 2{,}84 \end{bmatrix}.$$

Koska data oli jaettu opetus- ja testimatriiseihin, lasketaan testimatriisissa olevien alkioiden avulla RMSE, joksi saadaan

$$\mathrm{RMSE}\left(L'_{\overline{x};\gamma=0{,}2}\right) = \sqrt{\frac{1}{4}\left((3{,}23-2)^2 + \cdots + (1-2{,}31)^2\right)} = 1{,}3655\ldots.$$

Vastaavasti voitaisiin laskea myös MAE, ja se jätetään lukijalle harjoitustehtäväksi. Ilman regularisointikerrointa (tai asetettaessa $\gamma = 0$) on approksimaatiomatriisi muotoa

$$L'_{\overline{x}} = \begin{bmatrix} 3{,}80 & 4{,}29 & 4{,}91 & 3{,}26 & 1{,}10 \\ 1{,}31 & 1{,}61 & 1{,}96 & 3{,}14 & 3{,}51 \\ 2{,}99 & 3{,}37 & 3{,}86 & 2{,}49 & 0{,}75 \\ 1{,}63 & 1{,}95 & 2{,}32 & 2{,}99 & 2{,}94 \end{bmatrix},$$

ja vastaava RMSE

$$\mathrm{RMSE}\left(L'_{\overline{x}}\right) = \sqrt{\frac{1}{4}\left((3{,}26-2)^2 + \cdots + (1-2{,}32)^2\right)} = 1{,}3402\ldots.$$

Tällä kertaa valinta regularisointikertoimeksi ei tuottanut tulosta, ainakaan jos vertaa RMSE-arvoja keskenään. Myös itse approksimaatioita vertaillessa huomataan, että oikeastaan vain Bertan ensimmäiselle kauhuelokuvalle saatu approksimaatio on regularisointitermiä käyttäessä hieman lähempänä oikeaa arvostelua 5, mutta siihen se sitten jääkin. Itse asiassa käytössä olevalle testimatriisille kahden singulaariarvon tapauksessa parasta on jättää regularisointitermi kokonaan pois, ja virhe kasvaa kerrointa kasvatettaessa. Regularisointikertoimen toiminta onkin täysin datasta riippuvaista, sillä singulaariarvojen määrän ja numeeristen arvojen kasvaessa alkaa kertoimesta olemaan jo hyötyäkin.

Tarkastellaan regularisointikertoimen toimivuutta uudelleen oikealla elokuva-arvosteludatalla, mutta ennen sitä käydään vielä yksi idea approksimaation parantamiseksi.

Alkuarvauksen ja approksimaatiomatriisin uudelleenkokoamisen jälkeen voidaan kokeilla iterointimenetelmää, joka on esitelty kirjassa [**1**, kohta 3.6.5.1]. Iterointimenetelmän vaiheet ovat seuraavat:

1. Alustus: Alusta matriisin R jokaisen rivin i puuttuvat arvot kyseisen rivin alkioiden keskiarvolla $\overline{x}$, jolloin saadaan matriisi $R_{\overline{x}}$.
2. Iteratiivinen vaihe 1: Tee matriisille $R_{\overline{x}}$ valittua astetta k vastaava singulaariarvohajotelma $U_k \Sigma_k V_k^T$.
3. Iteratiivinen vaihe 2: Vaihda matriisista $R_{\overline{x}}$ alun perin puuttuneet arvot approksimaation $U_k \Sigma_k V_k^T$ antamiin arvoihin. Palaa iteratiiviseen vaiheeseen 1.

Iterointia voidaan jatkaa, kunnes puuttuvien arvojen approksimaatiot suppenevat johonkin. Tämä on kuitenkin etenkin suuria datamääriä käsitellessä työlästä, joten datasta riippuen järkevää voi olla toistaa iterointia esimerkiksi 5–10 kertaa ja katsoa tilannetta.

Iterointimenetelmän tarkoitus on siis yrittää parantaa saatavaa approksimaatiota päivittämällä alun perin rivikeskiarvoilla tehdyt alkuarvaukset kussakin iterointivaiheessa saadun singulaariarvohajotelman antamilla approksimaatioilla. Kokeillaan iterointia aiemmin käytettyjen matriisien R ja $L_{\overline{x}}$ avulla. Merkitään tästä eteenpäin iterointikierrosten määrää kirjaimella τ. Lasketaan iteroiden viisi kierrosta NumPya apuna käyttäen, jolloin saadaan

$$L'_{\overline{x};\tau=1} = \begin{bmatrix} 3{,}80 & 4{,}29 & 4{,}91 & 3{,}26 & 1{,}10 \\ 1{,}31 & 1{,}61 & 1{,}96 & 3{,}14 & 3{,}51 \\ 2{,}99 & 3{,}37 & 3{,}86 & 2{,}49 & 0{,}75 \\ 1{,}63 & 1{,}95 & 2{,}32 & 2{,}99 & 2{,}94 \end{bmatrix} \quad \hat{L}_{\overline{x};\tau=1} = \begin{bmatrix} 4 & 5 & 4 & 3{,}26 & 1 \\ 1 & 1{,}61 & 2 & 3{,}14 & 4 \\ 2{,}99 & 3{,}37 & 5 & 2 & 1 \\ 2 & 1 & 2{,}32 & 4 & 2{,}94 \end{bmatrix}$$

$$L'_{\overline{x};\tau=2} = \begin{bmatrix} 3{,}81 & 4{,}50 & 4{,}79 & 2{,}95 & 1{,}11 \\ 1{,}35 & 1{,}16 & 1{,}98 & 3{,}57 & 3{,}60 \\ 3{,}26 & 3{,}88 & 4{,}10 & 2{,}42 & 0{,}82 \\ 1{,}63 & 1{,}53 & 2{,}30 & 3{,}55 & 3{,}37 \end{bmatrix} \quad \hat{L}_{\overline{x};\tau=2} = \begin{bmatrix} 4 & 5 & 4 & 2{,}95 & 1 \\ 1 & 1{,}16 & 2 & 3{,}57 & 4 \\ 3{,}26 & 3{,}88 & 5 & 2 & 1 \\ 2 & 1 & 2{,}30 & 4 & 3{,}37 \end{bmatrix}$$

$$L'_{\overline{x};\tau=3} = \begin{bmatrix} 3{,}82 & 4{,}61 & 4{,}67 & 2{,}66 & 1{,}09 \\ 1{,}33 & 0{,}87 & 1{,}95 & 3{,}78 & 3{,}77 \\ 3{,}50 & 4{,}26 & 4{,}27 & 2{,}33 & 0{,}87 \\ 1{,}66 & 1{,}32 & 2{,}33 & 3{,}79 & 3{,}61 \end{bmatrix} \quad \hat{L}_{\overline{x};\tau=3} = \begin{bmatrix} 4 & 5 & 4 & 2{,}66 & 1 \\ 1 & 0{,}87 & 2 & 3{,}78 & 4 \\ 3{,}50 & 4{,}26 & 5 & 2 & 1 \\ 2 & 1 & 2{,}33 & 4 & 3{,}61 \end{bmatrix}$$

$$L'_{\overline{x};\tau=4} = \begin{bmatrix} 3{,}84 & 4{,}68 & 4{,}57 & 2{,}42 & 1{,}06 \\ 1{,}29 & 0{,}68 & 1{,}93 & 3{,}87 & 3{,}88 \\ 3{,}71 & 4{,}55 & 4{,}41 & 2{,}25 & 0{,}92 \\ 1{,}69 & 1{,}22 & 2{,}38 & 3{,}91 & 3{,}74 \end{bmatrix} \quad \hat{L}_{\overline{x};\tau=4} = \begin{bmatrix} 4 & 5 & 4 & 2{,}42 & 1 \\ 1 & 0{,}68 & 2 & 3{,}87 & 4 \\ 3{,}71 & 4{,}55 & 5 & 2 & 1 \\ 2 & 1 & 2{,}38 & 4 & 3{,}74 \end{bmatrix}$$

$$L'_{\overline{x};\tau=5} = \begin{bmatrix} 3{,}85 & 4{,}74 & 4{,}49 & 2{,}24 & 1{,}03 \\ 1{,}26 & 0{,}53 & 1{,}93 & 3{,}91 & 3{,}93 \\ 3{,}89 & 4{,}79 & 4{,}51 & 2{,}19 & 0{,}96 \\ 1{,}72 & 1{,}17 & 2{,}44 & 3{,}96 & 3{,}82 \end{bmatrix} \quad \hat{L}_{\overline{x};\tau=5} = \begin{bmatrix} 4 & 5 & 4 & 2{,}37 & 1 \\ 1 & 0{,}53 & 2 & 3{,}91 & 4 \\ 3{,}89 & 4{,}79 & 5 & 2 & 1 \\ 2 & 1 & 2{,}44 & 4 & 3{,}82 \end{bmatrix}.$$

Huomataan, että aluksi arviot muuttuvat nopeasti, ja iterointia jatkettaessa lukujen muutos hidastuu. Itse asiassa tässä esimerkissä jo noin 25 iterointikierroksen paikkeilla

arvot muuttuvat enää hyvin vähän, ja

$$\hat{L}_{\overline{x};\tau=25} = \left[\begin{array}{ccc|cc} 4 & 5 & 4 & 1{,}75 & 1 \\ 1 & -0{,}31 & 2 & 3{,}82 & 4 \\ 4{,}92 & 6{,}21 & 5 & 2 & 1 \\ 2 & 1 & 2{,}91 & 4 & 3{,}96 \end{array}\right].$$

Kuitenkaan näin monen iterointikierroksen tekeminen ei ole laskennallisesti järkevää, ja esimerkiksi Cecilian toisen komediaelokuvan approksimaatio kasvaa yli arvosteluasteikon, mikä luonnollisesti näkyy laskettaessa RMSE. Itse asiassa pienin RMSE saavutetaan neljännen iterointikierroksen kohdalla, ja

$$\begin{aligned} \mathrm{RMSE}\left(L'_{\overline{x};\tau=4}\right) &= 0{,}9573\ldots \\ \mathrm{RMSE}\left(L'_{\overline{x};\tau=5}\right) &= 0{,}9942\ldots \\ &\vdots \\ \mathrm{RMSE}\left(L'_{\overline{x};\tau=25}\right) &= 1{,}5779\ldots. \end{aligned}$$

Neljännen ja viidennen iterointikierroksen RMSE ovat jo huomattavasti pienempiä kuin ilman iterointia saatu $\mathrm{RMSE}\left(L'_{\overline{x}}\right) = 1{,}3402\ldots$, ja iteroinnin voidaan sanoa toimineen hyvin. Vertailtaessa matriiseja $\hat{L}_{\overline{x};\tau=1}$ ja $\hat{L}_{\overline{x};\tau=5}$ näyttää iterointi vaikuttavan oleellisesti saataviin approksimaatioihin, ja ne ovat paremmin linjassa testimatriisin T arvojen kanssa.

Kokeillaan vielä yhdistää regularisointinkertoimen käyttö iterointimenetelmän kanssa. Tapoja tähän on monia: regularisointikerroin voi olla käytössä vain ensimmäisellä kierroksella, yhtä samaa kerrointa voidaan käyttää jokaisella kierroksella tai kaikille kierroksille voidaan valita oma regularisointikerroin. Kuitenkin selkeyden ja laskennallisen yksinkertaisuuden vuoksi käytetään esimerkeissä samaa regularisointikerrointa jokaisella iterointikierroksella. Nyt siis joka iterointivaiheessa vähennetään singulaariarvoista regularisointikerroin γ ennen matriisin uudelleenluomista.

Käytetään kerrointa $\gamma = 0{,}14$ viidellä iterointikierroksella, jolloin saadaan

$$\hat{L}_{\overline{x};\gamma=0{,}14;\tau=5} = \left[\begin{array}{ccc|cc} 4 & 5 & 4 & 2{,}22 & 1 \\ 1 & 0{,}55 & 2 & 3{,}78 & 4 \\ 3{,}74 & 4{,}58 & 5 & 2 & 1 \\ 2 & 1 & 2{,}42 & 4 & 3{,}67 \end{array}\right],$$

$$\mathrm{RMSE}\left(\hat{L}_{\overline{x};\gamma=0{,}14;\tau=5}\right) = 0{,}9869\ldots.$$

Verrattaessa aiempaan ilman regularisointikerrointa saatuun approksimaatiomatriisiin $\hat{L}_{\overline{x};\tau=5}$ ovat ne hyvin lähellä toisiaan, mutta RMSE on regularisointikertoimen kanssa hieman pienempi. Tällaisilla pienilläkin eroilla voi kuitenkin olla paljon merkitystä tehtäessä oikeita elokuvasuosituksia oikeille käyttäjille, ja lähdetään seuraavaksi tutkimaan osaa aidosta elokuva-arvosteludatasta.

MovieLens-tietoaineistoja käytetään yleisesti suosittelujärjestelmien testaamiseen ja yleiseen testailemiseen. GroupLens Research -ryhmä on kerännyt elokuva-arvosteludataa MovieLens -sivuston käyttäjistä jo 1990-luvulta lähtien. Seuraavassa esimerkissä on käytetty MovieLens Latest Small -tietoaineistoa, joka on kirjoitushetkellä viimeksi päivitetty syyskuussa 2018, ja löytyy osoitteesta https://grouplens.org/datasets/movielens/. MovieLens-tietoaineistojen historiasta on kerrottu lisää artikkelissa [**7**].

MovieLens Latest Small sisältää 100836 elokuva-arvostelua 9724 eri elokuvasta, joita on arvostellut 610 eri käyttäjää. Arvosteluasteikko on $[0{,}5; 5]$ ja arvosteluja on annettu puolikkaan tarkkuudella. Data on jaettu `ratings.csv`-tiedostoon, joka sisältää käyttäjien antamat arvostelut eri elokuville, ja `movies.csv`-tiedostoon, joka yhdistää elokuvien tunnisteet niiden nimiin ja genreihin. Arvosteluista on myös saatavilla aikaleimat ja käyttäjien jättämät lyhyet kommentit elokuvista, mutta näitä ei tulla seuraavaksi käyttämään.

Seuraavat Python-ohjelmarivit on esitetty ilman kommenttirivejä, ja halutessaan lukija voi tutustua kokonaiseen toimivaan elokuvasuositteluohjelmaan kommenttirivien kera liitteessä A. Avataan tarvittavat tiedostot Pythonilla Pandas-kirjastoa hyödyntäen ja jaetaan data opetus- ja testijoukkoon scikit-learn-koneoppimiskirjaston avulla. Jako opetus- ja testijoukkoihin saattaa poistaa joiltain elokuvilta kaikki arvostelut, jolloin kyseisiä elokuvia ei enää joukosta löydy ollenkaan. Kun tämä on otettu huomioon, muunnetaan vielä opetusjoukko taulukoksi, jonka riveinä ovat eri käyttäjät ja sarakkeina elokuvat, ja alkioina käyttäjien antamat arvostelut elokuville.

```
import pandas as pd
from sklearn.model_selection import train_test_split
arvostelut = pd.read_csv('ml-latest-small/ratings.csv', sep=',',
                         usecols=['userId', 'movieId', 'rating'])
opetusjoukko_i, testijoukko_i = train_test_split(arvostelut.index,
                                                 test_size=0.25,
                                                 random_state=42)
opetusjoukko = arvostelut.copy()
opetusjoukko.loc[testijoukko_i, 'rating'] = np.nan
testijoukko = arvostelut.loc[testijoukko_i]
elokuvat = pd.read_csv('ml-latest-small/movies.csv', sep=',',
                       usecols=['movieId', 'title', 'genres'])
arvostelutaulukko = opetusjoukko.pivot(index='userId', columns='movieId',
                                       values='rating')
```

Katsotaan, miltä arvostelutaulukko näyttää:

```
>>> arvostelutaulukko
movieId   1       2       3       4   ...  193583  193585  193587  193609
userId                                 ...
1        4.0     NaN     4.0     NaN  ...     NaN     NaN     NaN     NaN
2        NaN     NaN     NaN     NaN  ...     NaN     NaN     NaN     NaN
3        NaN     NaN     NaN     NaN  ...     NaN     NaN     NaN     NaN
```

```
4       NaN     NaN     NaN     NaN  ...     NaN     NaN     NaN     NaN
5       NaN     NaN     NaN     NaN  ...     NaN     NaN     NaN     NaN
...     ...     ...     ...     ...  ...     ...     ...     ...     ...
606     2.5     NaN     NaN     NaN  ...     NaN     NaN     NaN     NaN
607     NaN     NaN     NaN     NaN  ...     NaN     NaN     NaN     NaN
608     2.5     2.0     2.0     NaN  ...     NaN     NaN     NaN     NaN
609     3.0     NaN     NaN     NaN  ...     NaN     NaN     NaN     NaN
610     5.0     NaN     NaN     NaN  ...     NaN     NaN     NaN     NaN

[610 rows x 9724 columns]
```

Huomataan, että arvostelutaulukko on hyvin harva, ja jako opetus- ja testijoukkoon harventaa sitä entisestään. Alustetaan nyt taulukko rivikeskiarvoilla ja otetaan puuttuvien alkioiden indeksit talteen seuraavien ohjelmarivien avulla.

```
import numpy as np
def alusta_rivikeskiarvoilla(arvostelutaulukko):
    R = arvostelutaulukko.values
    puuttuvat = np.isnan(R)
    rivikeskiarvot = arvostelutaulukko.mean(axis=1, skipna=True)
    arvostelutkeskiarvoilla = arvostelutaulukko.T.fillna(rivikeskiarvot).T
    R_x = arvostelutkeskiarvoilla.values
    return R_x, puuttuvat
arvostelumatriisi, puuttuvat = alusta_rivikeskiarvoilla(arvostelutaulukko)
```

Luodaan seuraavaksi funktio, joka laskee arvostelumatriisille kompaktin singulaariarvohajotelman ja päivittää puuttuvat arvot saatuihin approksimaatioihin. Funktio sisältää mahdollisuuden iterointiin ja regularisointikertoimen käyttämiselle.

```
def svd_puuttuvien_iterointi(R, puuttuvat, k=5, iteroi=1, gamma=0):
    R_i = R.copy()
    for _ in range(iteroi):
        U, Sigma, VT = np.linalg.svd(R_i, full_matrices=False)
        Sigma_reg = np.maximum(Sigma - gamma, 0)
        R_k = np.dot(U[:, :k], np.dot(np.diag(Sigma_reg[:k]), VT[:k, :]))
        R_i[puuttuvat] = R_k[puuttuvat]
    approksimaatiot = pd.DataFrame(R_i, index=arvostelutaulukko.index,
                                   columns=arvostelutaulukko.columns)
    return approksimaatiot
```

Ennen funktion käyttämistä tutkitaan hieman arvostelumatriisin singulaariarvoja. Singulaariarvojen suuruuksista voidaan piirtää kuvaaja seuraavasti.

```
import matplotlib.pyplot as plt
_, Sigma, _ = np.linalg.svd(arvostelumatriisi, full_matrices=False)
plt.figure(figsize=(6, 5))
plt.plot(Sigma, marker='x', linestyle='none', color='deeppink')
plt.title('Singulaariarvojen kuvaaja')
plt.xlabel('Indeksi')
```

```
plt.ylabel('Singulaariarvon suuruus')
plt.yscale('log')
plt.grid(True, which='major', linewidth=0.5)
plt.grid(which='minor', linestyle='--', linewidth=0.5)
plt.show()
```

Kuvassa 5.1 näkyy singulaariarvojen suuruus logaritmisella asteikolla. Huomataan, että ensimmäinen singulaariarvo on huomattavasti muita suurempi. Tulostetaan 50 ensimmäistä singulaariarvoa:

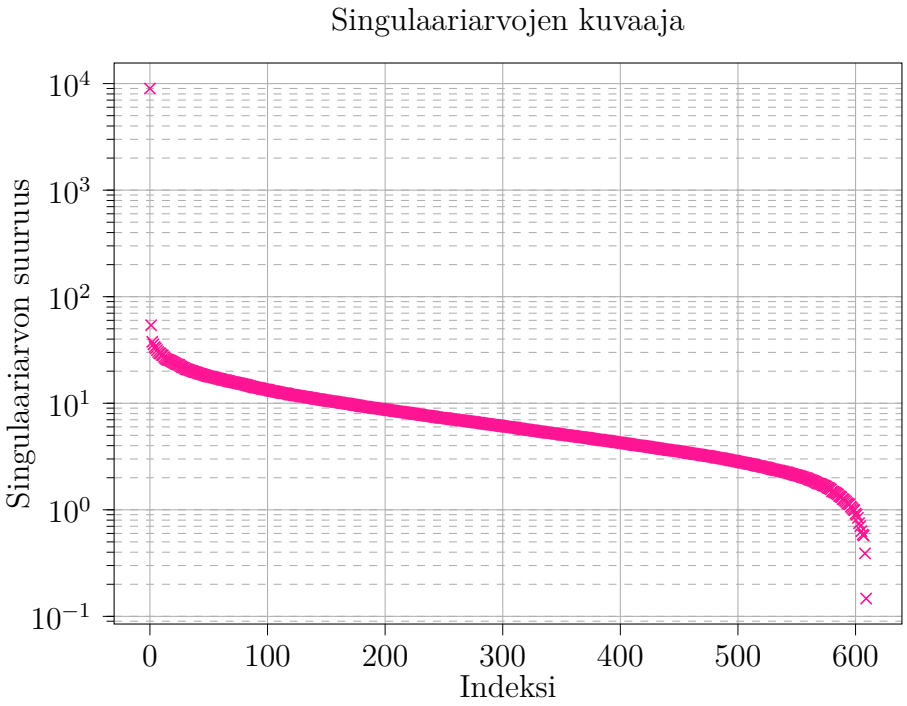
```
>>> Sigma[:50]
array([8992.21232335,   53.98001585,   37.70618249,   35.72806431,
         33.73512165,   32.85036841,   31.54076245,   30.31450433,
         29.40998868,   29.38378346,   28.4258151 ,   28.16493632,
         27.13647014,   26.41839386,   25.87507239,   25.57678697,
         25.48209036,   25.21600961,   24.87134497,   24.49924079,
         24.18054822,   23.89331423,   23.44470263,   23.21895399,
         23.00436662,   22.84752373,   22.3319844 ,   21.98379106,
         21.71119109,   21.51625511,   21.0329759 ,   20.85958047,
         20.72901496,   20.63047919,   20.3816032 ,   20.1509102 ,
         20.02609196,   19.79935621,   19.62089486,   19.57907412,
         19.40216521,   19.16291676,   18.97789951,   18.89701092,
         18.7672833 ,   18.60210916,   18.45282098,   18.3436594 ,
         18.08475187,   18.00085889])
```

Koska ensimmäinen singulaariarvo on huomattavan suuri suhteessa muihin, voi olla järkevää valita vain muutama singulaariarvo tämän lisäksi. Eri singulaariarvojen määriä on hyvä testailla ja toimivuutta vertailla keskenään. Kokeillaan valita 25 ensimmäistä singulaariarvoa, eli $k = 25$.

```
approksimaatiot = svd_puuttuvien_iterointi(arvostelumatriisi, puuttuvat,
                                           k=25)
```

Luodaan approksimaation tarkkuuden tarkastelua varten funktio, jonka avulla saadaan laskettua kätevästi RMSE ja MAE.

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
def laske_virheet(approksimaatiot, testijoukko):
    ennusteet = approksimaatiot.stack().reset_index()
    ennusteet.columns = ['userId', 'movieId', 'prediction']
    yhteenveto = pd.merge(testijoukko, ennusteet,
                          on=['userId', 'movieId'], how='inner')
    rmse = np.sqrt(mean_squared_error(yhteenveto['prediction'],
                                      yhteenveto['rating']))
    mae = mean_absolute_error(yhteenveto['prediction'],
                              yhteenveto['rating'])
    return rmse, mae
```



KUVA 5.1. MovieLens Latest Small -tietoaineistosta tehdyn opetusjoukon singulaariarvot visualisoituna kuvaajaan, kun aineistosta luodun matriisin tyhjät alkiot on korvattu kunkin käyttäjän omalla arvostelukeskiarvolla.

Lasketaan sitten RMSE ja MAE, kun approksimaatioon käytettiin 25 singulaariarvoa:

```
>>> laske_virheet(approksimaatiot, testijoukko)
(0.9245285790640793, 0.7140941127669529)
```

Datamatriisin harvuudesta huolimatta on päästy keskimäärin lähemmäksi oikeita arvoja kuin pienen esimerkkimatriisin tapauksessa. Tämä kuulostaa hyvältä, eikä vielä olla ehditty kokeilla regularisointikertoimen eikä iteroinnin vaikutusta. Luodaan kuitenkin ennen sitä funktio, joka mahdollistaa itse elokuvasuositusten tekemisen singulaariarvohajotelmaan perustuvaa approksimaatiota hyödyntäen. Kuten aiemmin sanottua, löytyvät tarkemmat koodin kommentit liitteestä A.

```
def suosittele(approksimaatiot, userId, elokuvat, arvostelut, suosit=20):
    u_ennusteet = approksimaatiot.loc[userId].sort_values(ascending=False)
    u_arvostelut = arvostelut[arvostelut.userId == userId]
    u_arvostelut = u_arvostelut.merge(elokuvat, how='left',
                                      left_on='movieId',
                                      right_on='movieId').sort_values(
                                          ['rating'], ascending=False)
    u_arvostelut = u_arvostelut[['movieId', 'rating', 'title', 'genres']]
    suositukset = elokuvat.merge(pd.DataFrame(u_ennusteet).reset_index(),
                                 how='left', left_on='movieId',
```

```
                                right_on='movieId').rename(
                                    columns={userId: 'prediction'})
    suositukset = suositukset[~suositukset['movieId'].isin(
        u_arvostelut['movieId'])]
    sarakkeet = ['movieId', 'prediction', 'title', 'genres']
    suositukset = suositukset[sarakkeet].sort_values(
        'prediction', ascending=False).head(suosit)
    u_arvostelut = u_arvostelut.set_index('movieId')
    suositukset = suositukset.set_index('movieId')
    return u_arvostelut, suositukset
```

Kokeillaan nyt suosittele-funktiota käyttäjälle, jonka id on 123.

```
kayttaja = 123
suosituksia = 20
arvosteltu, suositukset = suosittele(approksimaatiot, kayttaja, elokuvat,
                                     arvostelut, suosituksia)
```

Katsotaan ensin, mistä elokuvista käyttäjä on erityisesti pitänyt:

```
>>> arvosteltu.head(20)
         rating                          title                          genres
movieId
47          4.5   Seven (a.k.a. Se7en) (1995)              Mystery|Thriller
58803       4.5                     21 (2008)  Crime|Drama|Romance|Thriller
116797      4.5     The Imitation Game (2014)            Drama|Thriller|War
115569      4.5           Nightcrawler (2014)          Crime|Drama|Thriller
112556      4.5              Gone Girl (2014)                Drama|Thriller
111759      4.5       Edge of Tomorrow (2014)            Action|Sci-Fi|IMAX
109487      4.5           Interstellar (2014)                   Sci-Fi|IMAX
99114       4.5       Django Unchained (2012)          Action|Drama|Western
96829       4.5     Hunt, The (Jagten) (2012)                         Drama
85414       4.5            Source Code (2011) Action|Drama|Mystery|Sci-F...
80463       4.5    Social Network, The (2010)                         Drama
76093       4.5 How to Train Your Dragon (... Adventure|Animation|Childr...
260         4.5 Star Wars: Episode IV - A ...       Action|Adventure|Sci-Fi
68554       4.5        Angels & Demons (2009)  Crime|Drama|Mystery|Thriller
4226        4.5                Memento (2000)              Mystery|Thriller
2329        4.5     American History X (1998)                   Crime|Drama
2959        4.5             Fight Club (1999)   Action|Crime|Drama|Thriller
593         4.0 Silence of the Lambs, The ...         Crime|Horror|Thriller
104879      4.0              Prisoners (2013)        Drama|Mystery|Thriller
97923       4.0                 Flight (2012)                         Drama
```

Huomataan, että käyttäjä ei ole arvioinut yhtään elokuvaa viiden tähden arvoiseksi. Parhaat arvostelut saaneet elokuvat ovat olleet suurimmaksi osaksi trilleri-, draama- ja rikoselokuvia. Katsotaan 20 suositeltavaa elokuvaa käyttäjälle 123, kun approksimaatiossa käytettiin 25 singulaariarvoa:

```
>>> suositukset
     prediction                          title                          genres
movieId
356    4.173694           Forrest Gump (1994)      Comedy|Drama|Romance|War
457    4.095778          Fugitive, The (1993)                      Thriller
50     4.094587    Usual Suspects, The (1995)        Crime|Mystery|Thriller
608    4.092754                 Fargo (1996)   Comedy|Crime|Drama|Thriller
3578   4.090950              Gladiator (2000)        Action|Adventure|Drama
588    4.081821                Aladdin (1992) Adventure|Animation|Childr...
527    4.078624       Schindler's List (1993)                     Drama|War
1193   4.078098 One Flew Over the Cuckoo's...                         Drama
3593   4.076321      Battlefield Earth (2000)                 Action|Sci-Fi
4973   4.075809 Amelie (Fabuleux destin d'...                Comedy|Romance
150    4.074985              Apollo 13 (1995)          Adventure|Drama|IMAX
6377   4.074405           Finding Nemo (2003) Adventure|Animation|Childr...
3147   4.073661        Green Mile, The (1999)                    Crime|Drama
2012   4.072743 Back to the Future Part II... Adventure|Comedy|Sci-Fi|We...
2953   4.072722 Home Alone 2: Lost in New ...               Children|Comedy
2324   4.072498 Life Is Beautiful (La Vita...      Comedy|Drama|Romance|War
595    4.072444   Beauty and the Beast (1991) Animation|Children|Fantasy...
6934   4.070783 Matrix Revolutions, The (2... Action|Adventure|Sci-Fi|Th...
1      4.069051              Toy Story (1995) Adventure|Animation|Childr...
1035   4.068185    Sound of Music, The (1965)                Musical|Romance
```

Suositusten kärjessä on Forrest Gump (1994), mikä on yleisesti korkealle arvosteltu ja suosittu elokuva. Katsottaessa tarkemmin näyttävät suositukset sisältävän samojen genrejen elokuvia, kuin mitä käyttäjä oli itsekin arvioinut korkealle. Elokuvasuosittelut vaikuttavat oikein käyttökelpoisilta, ja kokeillaan seuraavaksi regularisointikertoimen vaikutusta saataviin suosituksiin.

Pidetään singulaariarvojen määrä samana $k = 25$, mutta vähennetään jokaisesta singulaariarvosta tarkkaan ja harkitusti valittu regularisointikerroin $\gamma = 9{,}3$. Pythonin komentotulkin kautta ajettuna saadaan määritettyä approksimaatio regularisointikertoimen kera, ja laskettua RMSE ja MAE:

```
>>> appr_g = svd_puuttuvien_iterointi(arvostelumatriisi, puuttuvat, k=25,
                                      gamma=9.3)
>>> laske_virheet(appr_g, testijoukko)
(0.9235819929186606, 0.7146493964843755)
```

Ero ilman regularisointikerrointa saatuihin virheisiin on erittäin pieni, eikä itse asiassa tällä singulaariarvojen määrällä paljoa tämän paremmaksi ilman iterointia päästäkään. Katsotaan kuitenkin huvin ja viihteen vuoksi regularisointikertoimen avulla saatavat elokuvasuositukset käyttäjälle id 123:

```
>>> _, suositukset_g = suosittele(appr_g, kayttaja, elokuvat,
                                  arvostelut, suosituksia)
>>> suositukset_g
```

```
     prediction                         title                         genres
movieId
356    4.189344           Forrest Gump (1994)      Comedy|Drama|Romance|War
50     4.113569    Usual Suspects, The (1995)        Crime|Mystery|Thriller
527    4.101814       Schindler's List (1993)                   Drama|War
457    4.101621          Fugitive, The (1993)                    Thriller
608    4.098064                 Fargo (1996)   Comedy|Crime|Drama|Thriller
3578   4.088624             Gladiator (2000)        Action|Adventure|Drama
110    4.087567            Braveheart (1995)              Action|Drama|War
1193   4.082470 One Flew Over the Cuckoo's...                        Drama
588    4.078374               Aladdin (1992) Adventure|Animation|Childr...
4973   4.077987 Amelie (Fabuleux destin d'...               Comedy|Romance
150    4.077894             Apollo 13 (1995)          Adventure|Drama|IMAX
6377   4.076121          Finding Nemo (2003) Adventure|Animation|Childr...
1210   4.075581 Star Wars: Episode VI - Re...       Action|Adventure|Sci-Fi
589    4.075504 Terminator 2: Judgment Day...                Action|Sci-Fi
1198   4.075190 Raiders of the Lost Ark (I...              Action|Adventure
1213   4.074973            Goodfellas (1990)                   Crime|Drama
364    4.074605        Lion King, The (1994) Adventure|Animation|Childr...
3147   4.073259       Green Mile, The (1999)                   Crime|Drama
858    4.073032        Godfather, The (1972)                   Crime|Drama
1      4.071299             Toy Story (1995) Adventure|Animation|Childr...
```

Saadut suositukset ovat melko samanlaisia ilman regularisointikerrointa saatujen kanssa, elokuvien keskinäinen järjestys on hieman muuttunut ja muutama arvostettu animaatioelokuva on ilmestynyt suositusten joukkoon.

Kokeillaan seuraavaksi iterointimenetelmää, ja lasketaan viisi iterointikierrosta:

```
>>> appr_i = svd_puuttuvien_iterointi(arvostelumatriisi, puuttuvat, k=25,
                                      iteroi=5)
>>> laske_virheet(appr_i, testijoukko)
(0.9242489031297572, 0.7052678277576947)
```

Itse asiassa kokeilemalla huomataan, että tällä kertaa iterointimenetelmä pelkiltään ei juurikaan paranna saatavia approksimaatioita, kun käytettiin 25 singulaariarvoa. Kolmanteen iterointikierrokseen asti RMSE ja MAE kyllä laskevat, mutta sen jälkeen ne lähtevät kasvamaan. Lukija voi kuitenkin halutessaan kokeilla iteroinnin toimivuutta ja tarkastella sillä saatavia elokuvasuosituksia. Pienemmällä singulaariarvojen määrällä voi myös päästä parempaan lopputulokseen, mutta puhutaan siitä vielä myöhemmin.

Otetaan nyt regularisointikerroin mukaan iterointiprosessiin, ja valitaan $\gamma = 15{,}6$. Nyt saadaan viidellä iterointikierroksella:

```
>>> appr_ig = svd_puuttuvien_iterointi(arvostelumatriisi, puuttuvat, k=25,
                                       iteroi=5, gamma=15.6)
>>> laske_virheet(appr_ig, testijoukko)
(0.8917441445093358, 0.6853268931176121)
```

Tulos on jo näkyvästi aiempia parempi, ja katsotaan nyt elokuvasuositusten laita samalle käyttäjälle id 123:

```
>>> suositukset_ig
     prediction                         title                         genres
movieId
356    4.422430           Forrest Gump (1994)      Comedy|Drama|Romance|War
50     4.317934    Usual Suspects, The (1995)        Crime|Mystery|Thriller
527    4.283024       Schindler's List (1993)                     Drama|War
457    4.270477          Fugitive, The (1993)                      Thriller
1198   4.256678 Raiders of the Lost Ark (I...              Action|Adventure
608    4.251614                 Fargo (1996)   Comedy|Crime|Drama|Thriller
110    4.247567            Braveheart (1995)              Action|Drama|War
912    4.221815            Casablanca (1942)                 Drama|Romance
1213   4.220838            Goodfellas (1990)                   Crime|Drama
1193   4.218896 One Flew Over the Cuckoo's...                         Drama
858    4.215946         Godfather, The (1972)                   Crime|Drama
1197   4.214117    Princess Bride, The (1987) Action|Adventure|Comedy|Fa...
6377   4.207480          Finding Nemo (2003) Adventure|Animation|Childr...
4973   4.206837 Amelie (Fabuleux destin d'...               Comedy|Romance
3578   4.199898             Gladiator (2000)        Action|Adventure|Drama
48516  4.197712          Departed, The (2006)          Crime|Drama|Thriller
1210   4.196780 Star Wars: Episode VI - Re...       Action|Adventure|Sci-Fi
364    4.194485         Lion King, The (1994) Adventure|Animation|Childr...
589    4.191826 Terminator 2: Judgment Day...                 Action|Sci-Fi
3147   4.183256        Green Mile, The (1999)                   Crime|Drama
```

Forrest Gump on edelleen suositeltavin elokuva, ja sen ennuste (engl. prediction) on noussut merkittävästi. Muistetaan, että käyttäjä oli alun perin antanut 17 elokuvalle arvosanan 4,5; eikä yksikään elokuva ollut saanut täyden viiden arvosanaa. On mielenkiintoista huomata, että elokuvien genret vastaavat hyvin pitkälti käyttäjän mieltymyksiä hänen antamiensa arvostelujensa perusteella, vaikka genrejä ei suositusten tuottamisessa hyödynnettykään.

Iteroinnin ja regularisoinnin avulla perinteinen SVD vaikuttaa toimivan melko hyvin suositusjärjestelmässä. Singulaariarvojen määrän, iterointikierrosten ja regularisaatiokertoimen vaikutusta kannattaa aina datasta riippuen kokeilla parempien suositusten mahdollistamiseksi, ja tämän aineiston tapauksessa pieni singulaariarvojen määrä näytti toimivan parhaiten johtuen ensimmäisen singulaariarvon suuruudesta. Valitsemalla $k \leq 10$ päästään ilman iterointia parempiin tuloksiin, mutta tällöin regularisointikertoimen vähentäminen singulaariarvoista ei paranna tuloksia eli pienin RMSE saadaan pitämällä $\gamma = 0$. Iterointikierrosten määrää kasvattamalla regularisointikertoimen hyöty kuitenkin nousee esiin, ja itse asiassa esimerkin määrällä $k = 25$ laskee RMSE vielä reilun 50 iterointikierrosten jälkeenkin, mutta tämä on laskennallisesti erittäin työlästä. Myös parhaan regularisointikertoimen löytäminen näin monen iterointikierroksen kokeiluissa on hankalaa. Lukija voi halutessaan kokeilla yhdistelmää $k = 25; \tau = 50; \gamma = 13{,}5$ ja katsoa, saisiko vielä parempia tuloksia aikaan toisenlaisilla yhdistelmillä.

Vuonna 2006 Netflix käynnisti kilpailun parhaan suosittelujärjestelmän löytämiseksi. Netflix tarjosi reilun 100 miljoonan arvostelun aineiston, ja suosittelujärjestelmien paremmuutta verrattiin luonnollisesti laskemalla RMSE. Netflix-kilpailuun liittyvien matriisihajotelmien kehityksestä on kerrottu lisää artikkelissa [**11**], ja mainitaan vielä nimeltä kaksi maininnan arvoista ratkaisua. Perinteisen singulaariarvohajotelman yksi iso ongelma on puuttuvien arvojen käsittely, sillä matriisin alustaminen vaikuttaa jo merkittävästi saataviin tuloksiin. *Funk SVD* approksimoi alkuperäistä matriisia R suoraan kahden matriisin tulona XY, ja käyttää gradienttimenetelmää matriisien X ja Y optimointiin. Ennustevirhe minimoidaan ainoastaan olemassa olevien arvojen perusteella. *SVD++* pohjautuu samaan ideaan, mutta se ottaa eksplisiittisen palautteen eli elokuva-arvostelujen lisäksi huomioon implisiittistä palautetta eli epäsuoraa käyttäytymistä käyttäjien ja elokuvien välillä, esimerkiksi selaushistoriaa tai katseluaikoja. Koska tämä tutkielma käsitteli singulaariarvohajotelman hyödyntämistä, jätetään Funk SVD ja SVD++ vain maininnan tasolle. Selkeyden vuoksi täytyy kuitenkin antaa seuraava huomautus.

Huomautus 5.4. Nimistään huolimatta Funk SVD ja SVD++ eivät pohjaudu singulaariarvohajotelmaan, vaan ovat täysin omia matriisihajotelmiaan. Funk SVD on esitelty Simon Funkin (nimimerkki, oikealta nimeltään Brandyn Webb) blogipostauksessa [**4**], ja SVD++ tekijänsä Yehuda Korenin julkaisussa [**10**].

Sekä oikea singulaariarvohajotelma, Funk SVD että SVD++ kärsivät uusien käyttäjien ja elokuvien ilmaantuessa *kylmäkäynnistysongelmasta* (engl. cold start problem). Esimerkiksi uutta käyttäjää ei voida suoraan vain lisätä jo olemassa olevaan datamatriisiin suositusten antamiseksi, vaan esimerkiksi SVD täytyy laskea kokonaan uudestaan. Toisaalta uudesta käyttäjästä ei ainakaan ihan heti ole ollenkaan informaatiota tarjolla, mitä voisi hyödyntää. Implisiittisen palautteen kerääminen onkin hyvä parannus, mutta myös SVD++ kärsii osittain kylmäkäynnistysongelmasta itse laskennan puolesta. Suuremmissa palveluissa suosituksiin liittyvät laskennat onkin järkevää tehdä usein erillään varsinaisesta palvelusta esimerkiksi öisin, ja tarjota uudelle käyttäjälle yleisluontoisia suosituksia aluksi. Nykyaikaiset syväoppimismenetelmät mahdollistavat entistä tarkempia ja toisaalta laskennallisesti työläämpiä tapoja suositusten toteuttamiseksi, mutta se on jo vähän *korkeampaa matematiikkaa*.

LIITE A

Python-ohjelmia

Luvuissa 4 ja 5 hyödynnetyt kokonaiset Python-ohjelmat perusteellisilla kommenteilla varustettuna. Ohjelmat löytyvät myös osoitteesta https://github.com/Tarmoboy/SVD-sovelluksia.

PYTHON-OHJELMA A.1. svd-kuvanpakkaus.py

```
'''
Tiedoston nimi: svd-kuvanpakkaus.py
Tekijä: Tarmo Ilves
Viimeksi muokattu: 16.5.2024
Kuvaus: Kuvanpakkausta singulaariarvohajotelman avulla. Ohjelma esittää
        tavan pakata värillisen kuvatiedoston pienempään kokoon.
'''
import numpy as np
import PIL.Image as img

def svd_kuvanpakkaus(kuvatiedosto, k):
    '''
    Singulaariarvohajotelmaan (SVD) perustuva tapa pakata värillinen kuva.

    Parametrit
    ----------
    kuvatiedosto : string
        Käsiteltävän kuvatiedoston sijainti string-muodossa.
    k : int
        Kuvanpakkauksessa käytettävien singulaariarvojen lukumäärä.

    Palauttaa
    ---------
    pakattu_kuva : numpy.ndarray
        Singulaariarvohajotelman avulla käsitelty kuva taulukkomuodossa.
    '''
    # Kuvan avaaminen
    kuva = img.open(kuvatiedosto)
    # Muunnos numpy-taulukoksi
    kuva_taulukkona = np.array(kuva)
    # Jaetaan värikanavat omiksi matriiseiksi R, G, B
    R = kuva_taulukkona[:, :, 0]
    G = kuva_taulukkona[:, :, 1]
    B = kuva_taulukkona[:, :, 2]
    # Käytetään singulaariarvohajotelmaa (SVD) eri värikanaville
    U_R, Sigma_R, VT_R = np.linalg.svd(R, full_matrices=False)
```

```
    U_G, Sigma_G, VT_G = np.linalg.svd(G, full_matrices=False)
    U_B, Sigma_B, VT_B = np.linalg.svd(B, full_matrices=False)
    # Käytetään k ensimmäistä singulaariarvoa ja lasketaan matriisitulo
    R_k = np.dot(U_R[:, :k], np.dot(np.diag(Sigma_R[:k]), VT_R[:k, :]))
    G_k = np.dot(U_G[:, :k], np.dot(np.diag(Sigma_G[:k]), VT_G[:k, :]))
    B_k = np.dot(U_B[:, :k], np.dot(np.diag(Sigma_B[:k]), VT_B[:k, :]))
    # Yhdistetään värikanavat yhtenäiseksi kuvaksi
    pakattu_kuva = np.stack([R_k, G_k, B_k], axis=-1)
    # Varmistetaan, että kaikkien pikselien arvot kuuluvat välille [0, 255]
    pakattu_kuva = np.clip(pakattu_kuva, 0, 255).astype(np.uint8)
    return pakattu_kuva

# Värikuvatiedoston sijainti
kuvatiedosto = 'kuva.jpg'
# Käytettävien singulaariarvojen lukumäärä
k = 250
# Funktion kutsuminen
pakattu_kuva = svd_kuvanpakkaus(kuvatiedosto, k)
# Muunnetaan saatu taulukko kuvaksi ja tallennetaan
tallennettava_kuva = img.fromarray(pakattu_kuva).save(f'svd_k{k}.jpg')
```

PYTHON-OHJELMA A.2. svd-elokuvasuosittelu.py

```
'''
Tiedoston nimi: svd-elokuvasuosittelu.py
Tekijä: Tarmo Ilves
Viimeksi muokattu: 17.5.2024
Kuvaus: Elokuvasuositusten tekeminen singulaariarvohajotelman avulla.
        Mahdollisuus myös iteroimalla sekä regularisointikertoimella
        parantaa saatavia suosituksia. Esimerkkinä toiminnasta käytetty
        MovieLens Latest Small -tietoaineistoa.
'''
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
#import matplotlib.pyplot as plt

def alusta_rivikeskiarvoilla(arvostelutaulukko):
    '''
    Muuttaa arvostelutaulukon matriisiksi ja täydentää sen puuttuvat arvot
    kunkin rivin keskiarvolla.

    Parametrit
    ----------
    arvostelutaulukko : pandas.DataFrame
        Taulukko, joka sisältää puuttuvia alkioita.

    Palauttaa
```

```
28      ---------
29      R_x : numpy.ndarray
30          Arvostelutaulukosta luotu matriisi, jonka puuttuvat alkiot on
31          täydennetty rivikeskiarvoilla.
32      puuttuvat : numpy.ndarray
33          Boolean-matriisi, joka kertoo puuttuneiden alkioiden indeksit.
34      '''
35      # Matriisimuotoon
36      R = arvostelutaulukko.values
37      # Puuttuvista arvoista boolean-matriisi
38      puuttuvat = np.isnan(R)
39      # Rivikeskiarvojen laskeminen
40      rivikeskiarvot = arvostelutaulukko.mean(axis=1, skipna=True)
41      # Rivikeskiarvojen syöttö taulukkoon
42      arvostelutkeskiarvoilla = arvostelutaulukko.T.fillna(rivikeskiarvot).T
43      # Taulukko matriisimutoon
44      R_x = arvostelutkeskiarvoilla.values
45      return R_x, puuttuvat
46
47  def svd_puuttuvien_iterointi(R, puuttuvat, k=5, iteroi=1, gamma=0):
48      '''
49      Laskee singulaariarvohajotelman (SVD) matriisille R ja päivittää vain
50      siitä puuttuvat arvot iteroimalla.
51
52      Parametrit
53      ----------
54      R : numpy.ndarray
55          Matriisi, jolle halutaan tehdä SVD.
56      puuttuvat : numpy.ndarray
57          Boolean-matriisi, joka sisältää tiedon alun perin matriisista R
58          puuttuneista alkioista.
59      k : int, valinnainen
60          Approksimaatiossa käytettävien singulaariarvojen lukumäärä.
61          Oletuksena k=5.
62      iteroi : int, valinnainen
63          Kuinka monta kertaa halutaan laskea puuttuvat arvot uusiksi.
64          Oletuksena lasketaan vain kerran.
65      gamma : float, valinnainen
66          Regularisointikerroin, joka vähennetään singulaariarvoista ennen
67          matriisin uudelleenluomista. Oletuksena kerrointa ei käytetä.
68
69      Palauttaa
70      ---------
71      approksimaatiot : pandas.DataFrame
72          Matriisin R approksimaatio taulukkomuodossa.
73      '''
74      R_i = R.copy()
75      for _ in range(iteroi):
76          # Kompakti SVD
77          U, Sigma, VT = np.linalg.svd(R_i, full_matrices=False)
```

```
78          # Regularisointikertoimen vähentäminen singulaariarvoista
79          Sigma_reg = np.maximum(Sigma - gamma, 0)
80          # Approksimaatio astetta k
81          R_k = np.dot(U[:, :k], np.dot(np.diag(Sigma_reg[:k]), VT[:k, :]))
82          # Vain puuttuvat arvot päivitetään
83          R_i[puuttuvat] = R_k[puuttuvat]
84      # Muunnos taulukoksi
85      approksimaatiot = pd.DataFrame(R_i, index=arvostelutaulukko.index,
86                                     columns=arvostelutaulukko.columns)
87      return approksimaatiot
88
89  def laske_virheet(approksimaatiot, testijoukko):
90      '''
91      Laskee RMSE ja MAE testijoukon ja ennustettujen arvostelujen välillä.
92
93      Parametrit
94      ----------
95      approksimaatiot : pandas.DataFrame
96          SVD:n avulla lasketut approksimaatiot puuttuneille alkioille.
97      testijoukko : pandas.DataFrame
98          Testidata, joka sisältää todelliset arvostelut.
99
100     Palauttaa
101     ---------
102     rmse : float
103         Laskettu keskineliövirheen neliöjuuri (RMSE).
104     mae : float
105         Laskettu keskimääräinen absoluuttinen virhe (MAE).
106     '''
107     # Muutetaan approksimaatiot-taulukosta rating -> prediction
108     ennusteet = approksimaatiot.stack().reset_index()
109     ennusteet.columns = ['userId', 'movieId', 'prediction']
110     # Yhdistetään testijoukko ja ennusteet
111     yhteenveto = pd.merge(testijoukko, ennusteet,
112                           on=['userId', 'movieId'], how='inner')
113     # Lasketaan virheet
114     rmse = np.sqrt(mean_squared_error(yhteenveto['prediction'],
115                                       yhteenveto['rating']))
116     mae = mean_absolute_error(yhteenveto['prediction'],
117                               yhteenveto['rating'])
118     return rmse, mae
119
120 def suosittele(approksimaatiot, userId, elokuvat, arvostelut, suosit=20):
121     '''
122     Antaa elokuvasuosituksia SVD-approksimaatioon perustuen.
123
124     Parametrit
125     ----------
126     approksimaatiot : pandas.DataFrame
127         SVD:n antamat approksimaatiot muutettuna taulukoksi.
```

```
128     userId : int
129         Käyttäjän userId, jolla tietoa etsitään taulukoista.
130     elokuvat : pandas.DataFrame
131         Tiedon elokuvista sisältävä taulukko, tarvittavat sarakkeet ovat
132         'movieId', 'title', 'genres'.
133     arvostelut : pandas.DataFrame
134         Alkuperäiset elokuva-arvostelut sisältä taulukko, tarvittavat
135         sarakkeet ovat 'userId', 'movieId', 'rating'.
136     suosit : int, valinnainen
137         Kuinka monta elokuvasuositusta halutaan palautettavan, oletus 20.
138 
139     Palauttaa
140     ---------
141     u_arvostelut : pandas.DataFrame
142         Kaikki valitun käyttäjän antamat elokuva-arvostelut.
143     suositukset : pandas.DataFrame
144         Elokuvasuosituksia valitulle käyttäjälle.
145     '''
146     # Ennustetut arvostelut käyttäjälle userId
147     u_ennusteet = approksimaatiot.loc[userId].sort_values(ascending=False)
148     # Käyttäjän jo olemassa olevat arvostelut
149     u_arvostelut = arvostelut[arvostelut.userId == userId]
150     u_arvostelut = u_arvostelut.merge(elokuvat, how='left',
151                                       left_on='movieId',
152                                       right_on='movieId').sort_values(
153                                           ['rating'], ascending=False)
154     # Tarpeeton userId -sarake pois
155     u_arvostelut = u_arvostelut[['movieId', 'rating', 'title', 'genres']]
156     # Ennusteiden yhdistäminen elokuvatietoihin
157     suositukset = elokuvat.merge(pd.DataFrame(u_ennusteet).reset_index(),
158                                  how='left', left_on='movieId',
159                                  right_on='movieId').rename(
160                                      columns={userId: 'prediction'})
161     # Jätetään pois käyttäjän jo arvostelemat elokuvat
162     suositukset = suositukset[~suositukset['movieId'].isin(
163         u_arvostelut['movieId'])]
164     # Sarakkeiden järjestys vastaamaan arvostelutaulukkoa
165     sarakkeet = ['movieId', 'prediction', 'title', 'genres']
166     # Suositukset suuruusjärjestykseen ja valitaan ensimmäiset
167     suositukset = suositukset[sarakkeet].sort_values(
168         'prediction', ascending=False).head(suosit)
169     # Tulostuksen yksinkertaistamiseksi indeksoinnit movieId mukaan
170     u_arvostelut = u_arvostelut.set_index('movieId')
171     suositukset = suositukset.set_index('movieId')
172     return u_arvostelut, suositukset
173 
174 # Käyttäjien elokuva-arvostelut sisältävän tiedoston luku
175 arvostelut = pd.read_csv('ml-latest-small/ratings.csv', sep=',',
176                          usecols=['userId', 'movieId', 'rating'])
177 # Arvosteludatan jakaminen opetus- ja testijoukkoihin 75 % - 25 %
```

```
opetusjoukko_i, testijoukko_i = train_test_split(arvostelut.index,
                                                 test_size=0.25,
                                                 random_state=42)
# Opetusjoukon taulukon koko vastaamaan arvostelut-taulukkoa
opetusjoukko = arvostelut.copy()
# Testijoukossa sijaitsevat arvostelut määrittelemättömiksi
opetusjoukko.loc[testijoukko_i, 'rating'] = np.nan
# Testijoukon luonti indeksien perusteella
testijoukko = arvostelut.loc[testijoukko_i]
# Elokuvatiedot sisältävän tiedoston luku
elokuvat = pd.read_csv('ml-latest-small/movies.csv', sep=',',
                       usecols=['movieId', 'title', 'genres'])
# Enemmän sarakkeita näkyviin
pd.set_option('display.max_columns', 8)
# Enemmän merkkejä tulostusleveyteen
pd.set_option('display.width', 150)
# Sarakkeille sopiva leveys gradua varten, muuta tarvittaessa
pd.set_option('display.max_colwidth', 30)
# Muunnos taulukoksi, jossa riveinä käyttäjät ja sarakkeina elokuvat
arvostelutaulukko = opetusjoukko.pivot(index='userId', columns='movieId',
                                       values='rating')
# Muunnos matriisiksi ja puuttuvien alkioiden etsiminen
arvostelumatriisi, puuttuvat = alusta_rivikeskiarvoilla(arvostelutaulukko)
# Singulaariarvojen piirtäminen kuvaajaan
#_, Sigma, _ = np.linalg.svd(arvostelumatriisi, full_matrices=False)
#plt.figure(figsize=(6, 5))
#plt.plot(Sigma, marker='x', linestyle='none', color='deeppink')
#plt.title('Singulaariarvojen kuvaaja')
#plt.xlabel('Indeksi')
#plt.ylabel('Singulaariarvon suuruus')
#plt.yscale('log')
#plt.grid(True, which='major', linewidth=0.5)
#plt.grid(which='minor', linestyle='-', linewidth=0.5)
#plt.show()
# Iteratiivinen SVD
approksimaatiot = svd_puuttuvien_iterointi(arvostelumatriisi, puuttuvat,
                                           k=25, iteroi=5, gamma=15.5)
# RMSE ja MAE laskeminen
rmse, mae = laske_virheet(approksimaatiot, testijoukko)
# Käyttäjän id:n valinta
kayttaja = 123
# Elokuvasuositusten lukumäärä
suosituksia = 20
# Suosittele-funktion käyttö
arvosteltu, suositukset = suosittele(approksimaatiot, kayttaja, elokuvat,
                                     arvostelut, suosituksia)
print(f'RMSE: {rmse}')
print(f' MAE: {mae}')
print('------------------------\n')
print(f'Käyttäjän id {kayttaja} parhaiten arvostelemat elokuvat, '
```

```
228         '20 ensimmäistä')
229 print('-----------------------------------------------------------------')
230 print(arvosteltu.head(20), '\n')
231 print(f'Käyttäjälle id {kayttaja} suositeltavia elokuvia, {suosituksia} '
232         'suositelluinta')
233 print('-----------------------------------------------------------------')
234 print(suositukset)
```

Kirjallisuutta

[1] Charu C. Aggarwal: *Recommender Systems: The Textbook.* Springer, 2016.

[2] Sudipto Banerjee, Anindya Roy: *Linear Algebra and Matrix Analysis for Statistics.* CRC Press, 2014.

[3] Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong: *Mathematics for Machine Learning.* Cambridge University Press, 2020.

[4] Simon Funk: *Netflix Update: Try This at Home.* 2006, https://sifter.org/simon/journal/20061211.html.

[5] Mustansar Ali Ghazanfar, Adam Prügel-Bennett: *The Advantage of Careful Imputation Sources in Sparse Data-Environment of Recommender Systems: Generating Improved SVD-based Recommendations.* Informatica, 2013, volyymi 37, sivut 61–92.

[6] Rafael C. Gonzalez, Richard E. Woods: *Digital Image Processing.* Neljäs laitos, Pearson, 2018.

[7] F. Maxwell Harper, Joseph A. Konstan: *The MovieLens Datasets: History and Context.* ACM Transactions on Interactive Intelligent Systems (TiiS), 2016, volyymi 5, numero 4, artikkeli 19, sivut 1–19.

[8] Peter Harrington: *Machine Learning in Action.* Manning, 2012.

[9] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen: *LoRA: Low-Rank Adaptation of Large Language Models.* International Conference on Learning Representations, 2022.

[10] Yehuda Koren: *Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model.* KDD '08: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, sivut 426–434.

[11] Yehuda Koren, Robert Bell, Chris Volinsky: *Matrix Factorization Techniques for Recommender Systems.* Computer, 2009, volyymi 42, numero 8, sivut 30–37.

[12] Steven J. Leon: *Linear Algebra with Applications.* Yhdeksäs laitos, Pearson, 2015.

[13] Chi-Kwong Li, Gilbert Strang: *An Elementary Proof of Mirsky's Low Rank Approximation Theorem.* Electronic Journal of Linear Algebra, 2020, volyymi 36, sivut 694–697.

[14] David Poole: *Linear Algebra: A Modern Introduction.* Kolmas laitos, Brooks/Cole, 2011.

[15] Gilbert Strang: *Linear Algebra and Learning from Data.* Wellesley-Cambridge Press, 2019.

[16] Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd: *Generalized Low Rank Models.* Foundations and Trends® in Machine Learning, 2016, volyymi 9, numero 1, sivut 1–118.