Modellering en simulatie
Practicum 2: Lagerangbenaderingen

Wim Michiels en Nick Vannieuwenhoven

24 oktober 2019

1 Praktische informatie

Het practicum wordt individueel opgelost. Je maakt een *gestructureerd* verslag met duidelijk **onderscheiden** componenten waarin je elk van de opgaven individueel beantwoordt, ook wanneer deze bestaat uit het schrijven van broncode. *De broncode neem je op in het verslag.*

Het verslag wordt uiterlijk **vrijdag 6 december 2019** om **16u00** ingediend. Een papieren versie van het verslag dient zich ten laatste op dat moment in de studentenbrievenbus in gebouw 200A te bevinden. Bovendien stuur je per elektronische post naar `nick.vannieuwenhoven@kuleuven.be` een gecomprimeerde map met daarin een elektronische versie van je verslag (in het pdf-formaat) alsook je Matlabbronbestanden. De elektronische versie geldt als de ingediende versie.

Zorg ervoor dat je bronbestanden de **opgelegde naamgeving** respecteren (zie verder) want de codes zullen automatisch geverifieerd worden. De naam van het verslag en de gecomprimeerde map dient overeen te stemmen met je studentennummer wanneer de bestandsextensie buiten beschouwing wordt gelaten; bijvoorbeeld "s0123456.pdf" voor je verslag en "s0123456.zip," "s0123456.tar.gz," enzoverder voor de gecomprimeerde map.

Veel succes!

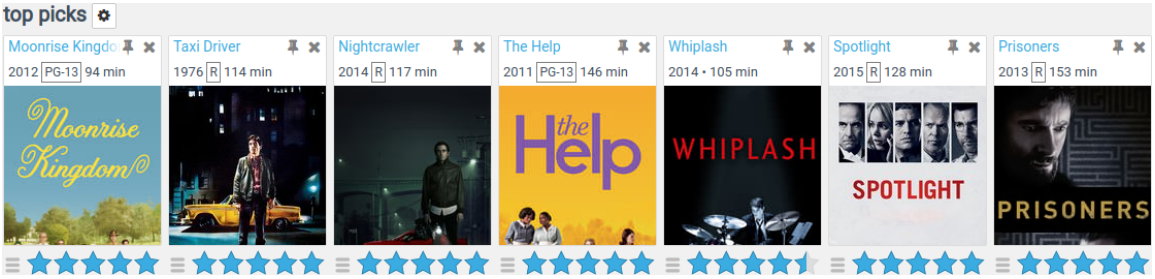
In wat volgt vervang je elk voorkomen van "SN" met jouw studentennummer (r-nummer)—bijvoorbeeld "r0123456". Elk van de functies dien je in een afzonderlijk bestand met dezelfde naam als de functie (met extensie ".m") te implementeren. Je codes zullen automatisch geverifieerd worden. Indien de opgelegde naamgeving niet gerespecteerd wordt, dan wordt dit beschouwd als het niet oplossen van de betrokken vraag.

Voor het oplossen van de opgaven mag je **alle** ingebouwde Matlabfuncties gebruiken. Naast de verplichte functies die de opgelegde naamgeving dienen te respecteren, mag je ook nog zelf extra hulpfuncties implementeren indien je dit nodig acht. Zorg ervoor dat deze hulproutines, voor zover ze in een eigen Matlabbronbestand geïmplementeerd worden, een naam dragen die start met "SN_".

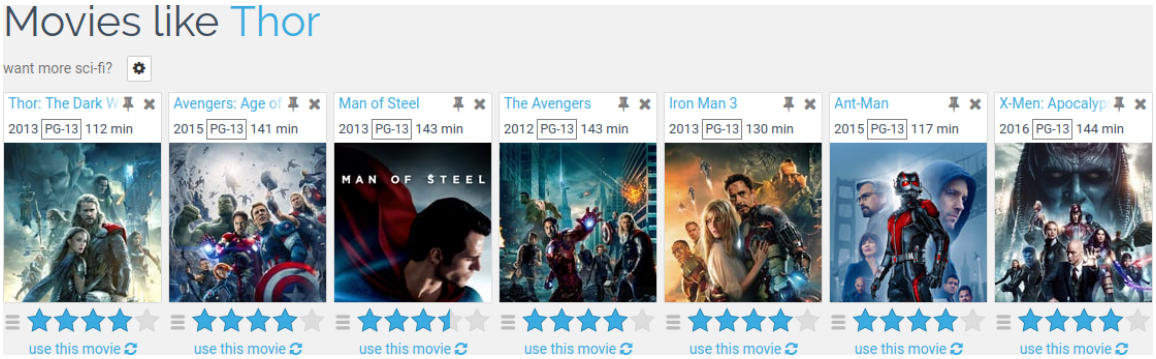
Houd **de tijd** die je aan dit practicum besteedt bij; zie opdracht 19.

2 Checklist bij het indienen

1. Broncode in het verslag opgenomen.
2. Elke opdracht afzonderlijk beantwoord.
3. Alle geïmplementeerde functies starten met je studentennummer.
4. De hoofding van elke functie zoals beschreven in de opdrachten wordt gerespecteerd.



Figuur 1: Beste keuzes voor een specifieke gebruiker volgens het aanbevelingssysteem van MovieLens.



Figuur 2: Films die lijken op *Thor* volgens het aanbevelingssysteem van MovieLens.

3 Een aanbevelingssysteem voor films

MovieLens (`https://movielens.org/`) is een niet-commercieel *collaborative filtering* systeem voor het aanbevelen van films op basis van beoordelingen die door de gebruikers van MovieLens werden ingevoerd. Voor het aanbevelen van films maakt MovieLens gebruik van een zogenaamd aanbevelingsysteem. Op basis van gebruikersbeoordelingen (op een schaal van 0.5 tot en met 5 sterren in stappen van 0.5 sterren) genereert dit systeem een gepersonaliseerde lijst van aanbevolen films die de gebruiker nog niet beoordeeld heeft. Een voorbeeld hiervan wordt weergegeven in fig. 1. Daarnaast kan het aanbevelingsysteem van MovieLens ook automatisch gelijkaardige films detecteren. In fig. 2 wordt een voorbeeld gegeven van de films die lijken op de film *Thor* uit 2011. De aanbevelingen die MovieLens in deze rubriek genereert zijn niet gepersonaliseerd: ze houden geen rekening met de films die je zelf eerder hebt beoordeeld.

In deze opgave zullen we een nieuw aanbevelingssysteem ontwikkelen voor MovieLens op basis van de gekende gebruikersbeoordelingen in hun publiek beschikbare databank. Het algoritme steunt enkel op eenvoudige principes uit de lineaire algebra en statistiek. De prestaties van het algoritme zullen beoordeeld worden op een kleine deelverzameling van de laatst beschikbare volledige databank van MovieLens, met name deze van september 2018. De volledige databank bevat ongeveer 27 miljoen beoordelingen van zo'n 280 000 geregistreerde gebruikers voor circa 58 000 films. Teneinde de geheugenvereisten voor dit practicum te beperken, zullen we met circa 6% van de volledige dataset werken. Deze beperktere dataset bevat maar 2 206 films en 12 488 gebruikers die tezamen 1 654 083 beoordelingen produceerden. De deelverzameling werd opgesteld door eerst de films te reduceren tot films die ten vroegste op 1 januari 2010 gepubliceerd werden, daarna werden alle films met strikt minder dan 75 beoordelingen verwijderd en tenslotte werden daarna alle gebruikers die strikt minder dan 50 films beoordeelden verwijderd.

In dit practicum zullen we een alternatief ontwikkelen voor het aanbevelingssysteem van MovieLens. Het ontwikkelde systeem zal dezelfde twee taken ondersteunen: gepersonaliseerde *top picks* en aanbeveling van gelijkaardige films. Deze twee componenten worden in de volgende delen uitgewerkt.

3.1 Matrixvervollediging

De beoordeling van een gebruiker voor een film kan op natuurlijke wijze voorgesteld worden door een matrix. We werken een voorbeeld uit. Veronderstel dat onze catalogus bestaat uit de films *John Wick* (2014), *Skyfall* (2012), *Inception* (2010), *Argo* (2012) en *The Great Gatsby* (2013). De beoordelingen van

11 willekeurige gebruikers uit MovieLens' databank zou men dan als volgt kunnen voorstellen:

$$R = \begin{array}{r} \text{John Wick} \\ \text{Skyfall} \\ \text{Inception} \\ \text{Argo} \\ \text{The Great Gatsby} \end{array} \begin{bmatrix} ? & ? & \mathbf{3.5} & \mathbf{3.0} & ? & \mathbf{3.0} & ? & ? & ? & \mathbf{3.0} & ? \\ \mathbf{2.5} & ? & \mathbf{4.0} & ? & ? & ? & ? & ? & \mathbf{5.0} & \mathbf{2.5} & ? \\ \mathbf{2.5} & \mathbf{5.0} & \mathbf{5.0} & \mathbf{5.0} & \mathbf{4.0} & \mathbf{4.0} & ? & \mathbf{0.5} & \mathbf{4.5} & ? & \mathbf{3.0} \\ ? & ? & \mathbf{4.5} & ? & ? & ? & \mathbf{3.5} & ? & ? & ? & \mathbf{3.5} \\ \mathbf{2.5} & ? & ? & ? & ? & ? & ? & ? & ? & \mathbf{2.5} & ? \end{bmatrix}. \quad (1)$$

Elke kolom van R geeft de beoordeling van één gebruiker voor enkele van de films en elke rij van R geeft de beoordelingen van enkele gebruikers voor één film. Veel waarden in bovenstaande matrix zijn onbekend, zoals wordt aangegeven met een "?". Men noemt R een onvolledige matrix.

Vooreerst zullen we een onvolledige beoordelingenmatrix R compact voorstellen door een *ijle matrix* (*sparse matrix*) in Matlab. We stellen hiervoor de onbekende waarden in de beoordelingenmatrix gelijk aan 0. Een ijle matrix R wordt dan compact voorgesteld door een lijst van de triplets $(i, j, r_{i,j})$ waarvoor $r_{i,j} \neq 0$. Het *ijlheidspatroon* (*sparsity pattern*) $\Omega \in \mathbb{N} \times \mathbb{N}$ van een ijle matrix R is de verzameling van koppels (i, j) waarvoor $r_{i,j} \neq 0$. De diagonaalmatrix $D = \text{diag}(-1, -2, -3, -4, -5)$ kan als volgt worden voorgesteld als een ijle matrix in Matlab:

```
>> D = spdiags((-1:-1:-5)', 0, 5, 5)
D =
   (1,1)       -1
   (2,2)       -2
   (3,3)       -3
   (4,4)       -4
   (5,5)       -5
```

De bovenstaande voorstellingswijze als triplets $(i, j, r_{i,j})$ heet het *coordinaatformaat*. In dit voorbeeld is het ijlheidspatroon $\Omega = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\}$. Merk op dat $r_{i,j} = 0$ voor alle $(i, j) \notin \Omega$. Dus `D(2,3)` geeft 0 als uitvoer in Matlab. Je kan de onderliggende datastructuur efficiënt opvragen via de `find` functie; bijvoorbeeld:

```
>> [i,j,v] = find(D)
i =             j =             v =
   1               1               -1
   2               2               -2
   3               3               -3
   4               4               -4
   5               5               -5
```

Hierbij geldt de relatie $r_{i_k,j_k} = v_k$ voor elke $k = 1, \ldots, \zeta$ waarbij ζ het aantal niet-nulwaarden is van R. In Matlab kan je het aantal niet-nulwaarden efficiënt opvragen via de functie `nnz`, bijvoorbeeld `nnz(D)` geeft 5 als resultaat.

Om gepersonaliseerde aanbevelingen te maken voor de gebruiker met volgnummer i zullen we trachten de beoordeling van deze gebruiker voor alle films uit de catalogus proberen te voorspellen. We kunnen dit interpreteren als het invullen van de i^{de} kolom van de matrix R. Duid het aantal films uit de catalogus aan met m en het aantal gebruikers aan met n. We zullen dan in dit project een zogenaamd *matrixvervolledigingsprobleem* oplossen om de onbekende waarden van de matrix $R \in \mathbb{R}^{m \times n}$ te voorspellen. Uiteraard dienen we hiertoe enkele veronderstellingen te maken—we zullen veronderstellen dat de beoordelingen van gebuikers niet geheel willekeurig zijn maar eerder een bepaald model volgen. Specifiek zullen we in dit practicum veronderstellen dat de gebruiker-beoordelingenmatrix R een kleine rang $r \ll \min\{m, n\}$ heeft: de beoordeling $r_{i,j}$ van gebruiker i voor film j voldoet—bij benadering—aan de relatie

$$r_{i,j} \approx f_{i,1} \cdot w_{j,1} + f_{i,2} \cdot w_{j,2} + \cdots + f_{i,r} \cdot w_{j,r} = \sum_{k=1}^{r} f_{i,k} \cdot w_{j,k},$$

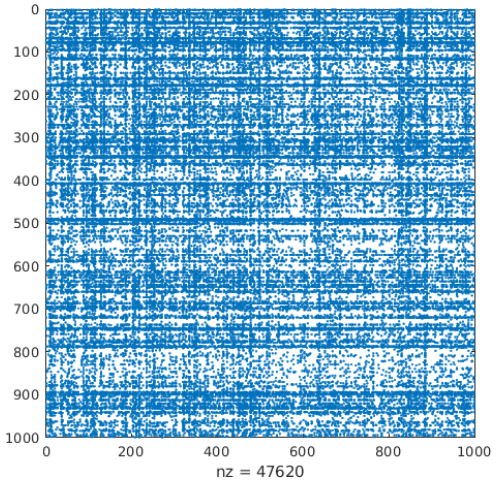
hetgeen equivalent is met

$$R \approx FW^T = \mathbf{f}_1\mathbf{w}_1^T + \mathbf{f}_2\mathbf{w}_2^T + \cdots + \mathbf{f}_r\mathbf{w}_r^T \quad (2)$$

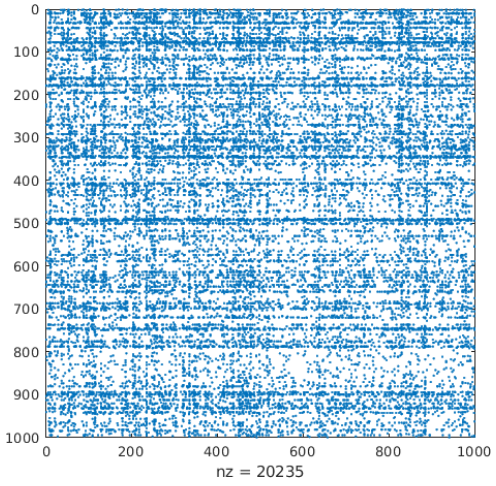
en waarbij $\mathbf{f}_i \in \mathbb{R}^m$, respectievelijk $\mathbf{w}_i \in \mathbb{R}^n$, de i^{de} kolom is van de matrix $F \in \mathbb{R}^{m \times r}$, respectievelijk $W \in \mathbb{R}^{n \times r}$. We kunnen model (2) als volgt interpreteren. Elke vector $\mathbf{f}_i$ kunnen we interpreteren als een

gewicht voor een bepaalde i^{de} feature of kenmerk van de film, terwijl $\mathbf{w}_i$ een voorliefde voorstelt van de gebruikers voor het i^{de} kenmerk. De featurevectoren $\mathbf{f}_i$ bevatten dus enkel informatie over de films, terwijl de voorliefdevectoren $\mathbf{w}_i$ enkel informatie bevatten over de gebruikers. Vanwege deze eigenschap spreekt men ook wel eens van een "gescheiden voorstelling." De veronderstelling dat R goed benaderd kan worden door een matrix van rang r impliceert dat de beoordeling van de meeste gebruikers grotendeels verklaard kan worden door r kenmerken van een film in combinatie met de voorliefde van de gebruikers voor de betrokken kenmerken. Zo zouden er bijvoorbeeld featurevectoren $\mathbf{f}_i$ kunnen zijn die een numerieke score toekennen aan de hoeveelheid actie, drama, humor, horror, intrige, romantiek en spanning voor elk van de films. Het is duidelijk dat er een objectief verschil bestaan in de hoeveelheid geweld in *The Hateful Eight* (2015) en de animatiefilm *Frozen* (2013). Een featurevector die de hoeveelheid geweld in elke film voorstelt, genoteerd als $\mathbf{f}_1$, zou dus zeker een grotere numerieke waarde bevatten voor de eerst vernoemde film dan voor de laatste. Het is eveneens duidelijk dat de gebruikers een verschillende voorliefde voor geweld kunnen bezitten. Deze verschillen kunnen door de voorliefdevector $\mathbf{w}_1$ voorgesteld worden; een positieve numerieke waarde duidt op een voorliefde voor geweld, een negatieve voor een afkeer en een waarde dicht bij 0 geeft indifferentie met betrekking tot geweld aan. De rang-1 matrix $\mathbf{f}_1\mathbf{w}_1^T$ bevat dan op positie (i, j) de bijdrage van het kenmerk "geweld" aan de totaalscore van gebruiker j voor film i.

Opdracht 1. Laad het bestand `MovieLens_Subset.mat` in. Dit bestand bevat de volgende elementen: `movieLabel` is een *cell array* die de titels bevat van de 2 206 films in de beperkte catalogus; `R` is de $2\,206 \times 12\,488$ onvolledige beoordelingenmatrix die 1 157 858 gekende beoordelingen bevat, compact voorgesteld als een ijle matrix; en `T` is eveneens een $2\,206 \times 12\,488$ onvolledige beoordelingenmatrix die 496 225 beoordelingen bevat, compact voorgesteld als een ijle matrix. De matrix `R` bevat de trainingsdata die gebruikt zal worden om de lagerangbenadering op te stellen, terwijl `T` de testdata bevat die we zullen gebruiken om de prestaties van het model te evalueren. Alle elementen van `R` en `T` die gelijk zijn aan 0 stellen een *onbekende* beoordeling voor. De i^{de} rij van `R` en `T` bevat de gekende beoordelingen van de film met titel `movieLabel{i}`. De j^{de} kolom van `R` en `T` bevat de gekende beoordelingen van de gebruiker met volgnummer j. Wanneer je `spy(R(1:1000,1:1000))` en `spy(T(1:1000,1:1000))` uitvoert, bekom je respectievelijk fig. 3 en fig. 4. ♦



Figuur 3: `spy(R(1:1000,1:1000))`



Figuur 4: `spy(T(1:1000,1:1000))`

Opdracht 2. Onderstel dat een geheel getal 4 bytes en een dubbele-precisie vlottendekommagetal 8 bytes in beslag neemt. Hoeveel geheugenruimte is nodig om de *volle* beoordelingenmatrix `full(R)` voor te stellen? Hoeveel geheugenruimte is nodig om de ijle beoordelingenmatrix `R` voor te stellen in het coordinaatformaat? Hoeveel geheugenruimte is vereist om de rang-r lagerangbenadering FW^T uit verg. (2) compact voor te stellen (als functie van r)? Maak een duidelijke figuur waarin je het geheugengebruik van de lagerangbenadering uit verg. (2) en het geheugengebruik van de ijle matrix `R` uitzet als functie van r voor de waarden $r = 1, \ldots, 500$. Neem deze figuur op in het verslag. Voor welke waarde van r snijden deze twee lineaire functies? ♦

Om de onbekende waarden in de matrix R te voorspellen, veronderstellen we dus dat R goed benaderd kan worden door een lagerangontbinding $R \approx FW^T$ zoals in verg. (2). Voor matrices *zonder* onbekende elementen kunnen we een lagerangontbinding op de volgende manier berekenen. Het welbekende Eckart–Young theorema stelt dat de optimale benadering van een matrix $X \in \mathbb{R}^{m \times n}$ met $m \leq n$ van rang $r \leq m$

gegeven wordt door de *afgeknotte singuliere waardenontbinding*. Laat de singuliere waardenontbinding van X gegeven zijn door

$$X = U \begin{bmatrix} \Sigma & 0 \end{bmatrix} V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 & 0 \\ 0 & \Sigma_2 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \\ V_3^T \end{bmatrix}$$

waarbij $U \in \mathbb{R}^{m\times m}$, $\Sigma \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, $U_1 \in \mathbb{R}^{m\times r}$, $\Sigma_1 \in \mathbb{R}^{r\times r}$ en $V_1 \in \mathbb{R}^{n\times r}$; de overige matrices hebben dimensies die compatibel zijn met de partitionering in bovenstaande vergelijking. De matrices U en V zijn orthogonale matrices en Σ is een diagonaalmatrix met positieve getallen op de diagonaal die aflopend gesorteerd zijn. Het Eckart–Young theorema stelt dan dat de beste rank-r benadering van X gegeven wordt door de singuliere waardenontbinding van X af te knotten na r termen:

$$\min_{\text{rank}(B)\leq r} \|X - B\|_F = \|X - U_1\Sigma_1 V_1^T\|_F.$$

Helaas kunnen we niet zonder meer een afgeknotte singuliere waardenontbinding berekenen van de onvolledige matrix R om tot een (optimale) benadering zoals in verg. (2) te komen. Bovendien is de matrix R bijzonder groot zodat het berekenen van een singuliere waardenontbinding tijdrovend is. Gelukkig bestaan er wel speciale algoritmen om de afgeknotte singuliere waardenontbinding van een ijle matrix efficiënt te berekenen. In Matlab kan dit met de functie `svds`. Zo levert `[U, S, V] = svds(R,5)` bijvoorbeeld de factoren van de singuliere waardenontbinding van de beste rang-5 benadering van de ijle matrix `R`, i.e., `U*S*V'` is de rang-5 afgeknotte singuliere waardenontbinding van `R`. Op basis van zulke algoritmen ontwierpen Wang et al. (2014)[1] een efficiënt iteratief algoritme om een goede—doch niet optimale—lagerangbenadering op te stellen van een onvolledige matrix.

3.2 Het aanbevelingsalgoritme

Om de werking van het algoritme van Wang et al. voor onvolledige matrices beter te begrijpen, werken we eerst het standaard *successive rank-1 deflation* algoritme uit voor het (inefficiënt) berekenen van een afgeknotte singuliere waardenontbinding van een volle matrix A. Deze methode wordt gegeven in algoritme 1. In elke iteratie berekent het algoritme een beste rang-1 benadering van E_j (regel 4) en trekt deze vervolgens van E_j af om aldus de volgende matrix E_{j+1} te genereren (regel 5).

Algorithm 1: Een iteratief algoritme voor het berekenen van een afgeknotte singuliere waardenontbinding van een volle matrix $A \in \mathbb{R}^{m\times n}$.

1 $E_0 \leftarrow A$;
2 $j \leftarrow 1$;
3 **while** $\|E_{j-1}\|_2 \neq 0$ **do**
4 | Zij $\sigma_j \geq 0$, $\mathbf{u}_j \in \mathbb{R}^m$ en $\mathbf{v}_j \in \mathbb{R}^n$ zodanig dat $\|E_{j-1} - \sigma_j \mathbf{u}_j \mathbf{v}_j^T\|_F = \min_{\text{rank}(B)\leq 1} \|E_{j-1} - B\|_F$;
5 | $E_j \leftarrow E_{j-1} - \sigma_j \mathbf{u}_j \mathbf{v}_j^T$;
6 | $j \leftarrow j + 1$;
7 **end**

Opdracht 3. Zij $A \in \mathbb{R}^{m\times n}$ een matrix van rang $r \leq \min\{m, n\}$ met singuliere waardenontbinding $A = \sum_{i=1}^{r} \gamma_i \mathbf{g}_i \mathbf{h}_i^T$. Je mag veronderstellen dat de singuliere waardenontbinding van A uniek is. Onderstel dat algoritme 1 op A wordt toegepast. Bewijs dat

$$A - E_k = \sum_{j=1}^{k} \gamma_j \mathbf{g}_j \mathbf{h}_j^T$$

de rang-k (met $k \leq r$) afgeknotte singuliere waardenontbinding van A is. Na hoeveel stappen stopt het algoritme? ♦

Het toepassen van algoritme 1 op een *ijle* matrix is geen goed idee, zelfs niet wanneer we slechts een beperkt aantal iteraties zouden uitvoeren.

[1] Wang et al., *Rank-one matrix pursuit for matrix completion*, JMLR: Workshop and Conference Proceedings, 32, 2014.

Opdracht 4. Veronderstel dat het berekenen van $(\alpha, \mathbf{x}, \mathbf{y})$ die de beste rang-1 benadering $\alpha\mathbf{x}\mathbf{y}^T$ voorstelt van een $m \times n$ matrix A slechts $m + n + 1$ dubbele-precisie vlottendekommagetallen (van 8 byte) vereist (om het resultaat voor te stellen). Zij c het laatste cijfer uit je studentennummer. Wat is c? Stel dat we de stappen 4–6 van algoritme 1 slechts $(c + 1)$ maal zouden uitvoeren. Kan men dit algoritme dan toepassen op een $280\,000 \times 58\,000$ matrix met 27 miljoen niet-nulwaarden (i.e., de volledige MovieLens databank) op een desktop met 8GB werkgeheugen en $2 + (c + 1)^2$ (in GB) swap geheugen? Verklaar waarom wel of niet? ♦

Om dit probleem op te lossen zal het algoritme van Wang et al. ervoor zorgen dat het ijlheidspatroon van E_j in elke iteratie gelijk is aan het ijlheidspatroon Ω van de invoer A. Deze aanpassing introduceert weliswaar een extra complicatie, namelijk dat de eenvoudige update in stap 5 van algoritme 1 niet meer optimaal is in de volgende zin. Beschouw het volgende optimalisatieprobleem

$$\min_{\mathbf{s}\in\mathbb{R}^k} \left\| A - \sum_{j=1}^{k} s_j \mathbf{u}_j \mathbf{v}_j^T \right\|_F . \tag{3}$$

Wanneer algoritme 1 wordt toegepast op een *volle* matrix, dan kan men aantonen dat de oplossing van bovenstaand probleem de vector

$$\begin{bmatrix}\sigma_1 & \sigma_2 & \cdots & \sigma_k\end{bmatrix}^T \tag{4}$$

is, waarbij σ_i de i^{de} grootste singuliere waarde van A is en $\mathbf{u}_i$ en $\mathbf{v}_i$ respectievelijk de bijbehorende linkse en rechtse singuliere vectoren zijn. Het idee van Wang et al. bestaat er uit om stap 5 uit algoritme 1 te vervangen door de volgende twee stappen:

$$5a: \quad \text{Zij } \mathbf{s}_k \in \mathbb{R}^k \text{ zodanig dat } \left\| A - \sum_{j=1}^{k} (\mathbf{s}_k)_j \cdot P_\Omega(\mathbf{u}_j \mathbf{v}_j^T) \right\|_F = \min_{\mathbf{z}\in\mathbb{R}^k} \left\| A - \sum_{j=1}^{k} z_j \cdot P_\Omega(\mathbf{u}_j \mathbf{v}_j^T) \right\|_F \tag{5}$$

$$5b: \quad E_k \leftarrow A - \sum_{j=1}^{k} (\mathbf{s}_k)_j \cdot P_\Omega(\mathbf{u}_j \mathbf{v}_j^T) \tag{6}$$

waarbij $Y = P_\Omega(X)$ de lineaire operator is die alle elementen buiten het ijlheidspatroon Ω van A gelijk stelt aan 0:

$$y_{i,j} = \begin{cases} x_{i,j} & \text{als } (i,j) \in \Omega \\ 0 & \text{anders} \end{cases} .$$

Het is eenvoudig te bewijzen dat P_Ω lineair is in zijn argument: $P_\Omega(\alpha X + \beta Y) = \alpha P_\Omega(X) + \beta P_\Omega(Y)$. We kunnen stap 5b dan ook beknopter schrijven als $E_k \leftarrow A - P_\Omega(U_k \operatorname{diag}(\mathbf{s}_k) V_k^T)$ waarbij de j^{de} kolom van U_k, respectievelijk V_k, de vector $\mathbf{u}_j \in \mathbb{R}^m$, respectievelijk $\mathbf{v}_j \in \mathbb{R}^n$, is.

Opdracht 5. Schrijf een functie met de hoofding

```
function [X] = SN_sparseModel(Uk,sk,Vk,A)
```

die als uitvoer `X` de matrix $X = P_\Omega(U_k \operatorname{diag}(\mathbf{s}_k) V_k^T)$ geeft. De invoer van de functie is een $m \times k$ matrix U_k, een kolomvector $\mathbf{s}_k \in \mathbb{R}^k$, een matrix $V \in \mathbb{R}^{n\times k}$ en een ijle matrix A met ijlheidspatroon Ω. Duid het aantal niet-nul elementen van A aan met ζ.

De geheugencomplexiteit van je algoritme[2] in functie van m, n, r en ζ mag hoogstens $\mathcal{O}(\zeta)$ bedragen—je mag hierbij wel veronderstellen dat $\max\{m, n\} \leq \zeta$. Hoe heb je deze doelstelling bereikt? ♦

We dienen ook het optimalisatieprobleem in verg. (5) op te lossen. Duid met $\operatorname{vec}(A)$ de kolomsgewijze vectorizatie van de matrix $A \in \mathbb{R}^{m\times n}$ aan:

$$\text{als } A = \begin{bmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_n\end{bmatrix} \text{ dan is } \operatorname{vec}(A) := \begin{bmatrix}\mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n\end{bmatrix} .$$

[2]De geheugencomplexiteit van de invoer en uitvoer tel je niet mee!

In Matlab kan de vectorizatie van een matrix `A` zeer eenvoudig berekend worden door `A(:)` of via de `reshape` functie. Merk op dat $\|A\|_F = \|\operatorname{vec}(A)\|_F = \|\operatorname{vec}(A)\|$. Bijgevolg is het optimalisatieprobleem in verg. (5) equivalent met het volgende kleinste-kwadratenprobleem:

$$\min_{\mathbf{z}\in\mathbb{R}^k} \left\| \operatorname{vec}(A) - \sum_{j=1}^{k} z_j \operatorname{vec}\big(P_\Omega(\mathbf{u}_j\mathbf{v}_j^T)\big) \right\| = \min_{\mathbf{z}\in\mathbb{R}^k} \|\mathbf{a} - B\mathbf{z}\|, \tag{7}$$

waarin $\mathbf{a} = \operatorname{vec}(A) = \operatorname{vec}(P_\Omega(A)) \in \mathbb{R}^{mn}$ en $B \in \mathbb{R}^{mn\times k}$ als j^{de} kolom $\operatorname{vec}\big(P_\Omega(\mathbf{u}_j\mathbf{v}_j^T)\big)$ heeft.

Opdracht 6. Zij $U_{15} \in \mathbb{R}^{m\times 15}$ en $V_{15} \in \mathbb{R}^{n\times 15}$ matrices wiens elementen gelijk zijn aan 1. Duid het ijlheidspatroon van `R` uit opdracht 1 aan met Ω. Bereken bovenstaande matrix $B \in \mathbb{R}^{mn\times 15}$ wiens i^{de} kolom $\operatorname{vec}(P_\Omega(\mathbf{u}_i\mathbf{v}_i^T))$ is. Plot de principale 500×15 deelmatrix door `spy(B(1:500,:))` uit te voeren. Neem de figuur op in het verslag. Welk patroon valt je op? Verklaar. ♦

Opdracht 7. Schrijf een functie met de hoofding

```
function [s] = SN_optimalCoefficients(Uk,Vk,A)
```

die als uitvoer $\mathbf{s}_k \in \mathbb{R}^k$ de optimale oplossing van het kleinste-kwadratenprobleem in verg. (7) berekent. De matrix $U_k \in \mathbb{R}^{m\times k}$ heeft als i^{de} kolom $\mathbf{u}_i$ zoals in verg. (7). De matrix $V_k \in \mathbb{R}^{n\times k}$ heeft als i^{de} kolom $\mathbf{v}_i$ zoals in verg. (7). De ijle matrix $A \in \mathbb{R}^{m\times n}$ is zoals in verg. (7) met ijlheidspatroon Ω en ζ niet-nul elementen. Je mag veronderstellen dat $k \leq mn$ en dat de matrix B uit verg. (7) een volle kolomrang heeft. Het is aanbevolen om in je implementatie gebruik te maken van de functie `SN_sparseModel`.

De geheugencomplexiteit van je algoritme in functie van m, n, k en ζ mag hoogstens $\mathcal{O}(k\zeta)$ bedragen—je mag hierbij wel veronderstellen dat $\max\{m, n\} \leq \zeta$. Hoe heb je deze doelstelling bereikt? ♦

Tot slot is er nog een extra ingrediënt noodzakelijk. Het is raadzaam om het algoritme van Wang et al. toe te passen op gecentreerde data. Aangezien de eerste taak van het aanbevelingsysteem eruit bestaat om gepersonaliseerde aanbevelingen te maken, opteren we ervoor om de ijle beoordelingenmatrix R te centreren door voor elke gebruiker i de beoordeling $r_{i,j}$ aan te passen volgens de formule $r_{i,j} \leftarrow r_{i,j} - \mu_i$, waarbij μ_i het gemiddelde is van alle gekende beoordelingen voor gebruiker i. Dit heeft als effect dat de individuele verschillen in de beoordelingsstijl van de gebruikers getemperd worden.

Opdracht 8. Schrijf een functie met de hoofding

```
function [mu] = SN_userMeans(A)
```

die als uitvoer een kolomvector `mu` van lengte n geeft waarvan het i^{de} element μ_i gelijk is aan het gemiddelde van alle *gekende* beoordelingen van gebruiker i. De invoermatrix $A \in \mathbb{R}^{m\times n}$ is een ijle matrix die de gekende beoordelingen bevat, bijvoorbeeld zoals de matrices `R` en `T` die je in opdracht 1 ingeladen hebt. ♦

Opdracht 9. Hoeveel bedragen de 3 laagste gemiddelde beoordelingen? Hoeveel gebruikers gaven exact 5 als gemiddelde score? ♦

We hebben nu alle onderdelen van het *rank-1 matrix pursuit* algoritme van Wang et al. besproken. Voor de volledigheid wordt het in algoritme 2 weergegeven. In dit algoritme is $\mathbf{1} \in \mathbb{R}^n$ de vector gevuld met enen. In alle volgende experimenten mag je gebruik maken van de Matlabfunctie `SN_rank1MatrixPursuit` in appendix A die algoritme 2 implementeert. Als eerste invoerargument verwacht het algoritme een ijle matrix $R \in \mathbb{R}^{m\times n}$ die de gekende beoordelingen voorstelt. Het tweede invoerargument is de gewenste rang $1 \leq k \leq \min\{m, n\}$ van de lagerangbenadering zoals in verg. (2). Het derde invoerargument T is een andere ijle $m \times n$ matrix die gekende beoordelingen voorstelt. De eerste drie uitvoerargumenten stellen de lagerangbenadering van R voor: $R \approx R_k := U_k \operatorname{diag}(\mathbf{s}_k) V_k^T$, waarbij $U_k \in \mathbb{R}^{m\times k}$, $\mathbf{s}_k \in \mathbb{R}^k$ en $V_k \in \mathbb{R}^{n\times k}$. Een precieze voorstelling zoals in verg. (2) bekom je door bijvoorbeeld $W = U_k \operatorname{diag}(\mathbf{s}_k)$ en $F = V_k$ te kiezen. Het laatste uitvoerargument is een kolomvector van lengte k die op positie j de RMSE (zie verder) bevat tussen de onvolledige matrix T en $P_\Omega(U_j \operatorname{diag}(\mathbf{s}_j) V_j^T) = P_\Omega(R_j)$, waarbij Ω het ijlheidspatroon van T is.

Wanneer we algoritme 2 toepassen op de matrix R uit verg. (1) met $k = 3$, dan vinden we, na afronding tot op één cijfer na de komma, de volgende voorspelling voor de onbekende waarden:

$$R \approx R_3 = \begin{array}{r} \text{John Wick} \\ \text{Skyfall} \\ \text{Inception} \\ \text{Argo} \\ \text{The Great Gatsby} \end{array} \begin{bmatrix} 2.5 & 5.0 & \mathbf{3.6} & \mathbf{3.0} & 4.0 & \mathbf{3.0} & 3.5 & 0.5 & 4.8 & \mathbf{2.9} & 3.4 \\ \mathbf{2.5} & 5.0 & \mathbf{3.9} & 3.9 & 4.0 & 3.4 & 3.5 & 0.5 & \mathbf{5.0} & \mathbf{2.4} & 3.1 \\ \mathbf{2.5} & \mathbf{5.0} & \mathbf{5.0} & \mathbf{5.0} & \mathbf{4.0} & \mathbf{4.0} & 3.5 & \mathbf{0.5} & \mathbf{4.6} & 2.5 & \mathbf{3.1} \\ 2.5 & 5.0 & \mathbf{4.5} & 4.1 & 4.0 & 3.5 & \mathbf{3.5} & 0.5 & 4.6 & 2.8 & \mathbf{3.3} \\ \mathbf{2.5} & 5.0 & 4.2 & 4.0 & 4.0 & 3.5 & 3.5 & 0.5 & 4.8 & \mathbf{2.6} & 3.2 \end{bmatrix},$$

Algorithm 2: Rank-1 matrix pursuit voor het vervolledigen van een onvolledige matrix $R \in \mathbb{R}^{m\times n}$.

$E_0 \leftarrow R$;
$\mathbf{u}_1 \leftarrow \mathbf{1}$;
$\mathbf{v}_1 \leftarrow$ `SN_userMeans(R)`;
$E_1 \leftarrow R - P_\Omega(\mathbf{u}_1\mathbf{v}_1^T)$;
$U_1 \leftarrow \mathbf{u}_1$;
$V_1 \leftarrow \mathbf{v}_1$;
$\mathbf{s}_1 \leftarrow [1]$;
for $j = 2, 3, \ldots, k$ **do**
  Zij $\sigma_j \geq 0$, $\mathbf{u}_j \in \mathbb{R}^m$ en $\mathbf{v}_j \in \mathbb{R}^n$ zodanig dat $\|E_{j-1} - \sigma_j\mathbf{u}_j\mathbf{v}_j^T\|_F = \min_{\mathrm{rank}(B)\leq 1}\|E_{j-1} - B\|_F$;
  Zij $\mathbf{s}_j \in \mathbb{R}^j$ zodanig dat $\left\|R - \sum_{i=1}^j (\mathbf{s}_j)_i \cdot P_\Omega(\mathbf{u}_i\mathbf{v}_i^T)\right\|_F = \min_{\mathbf{z}\in\mathbb{R}^j}\left\|R - \sum_{i=1}^j z_i \cdot P_\Omega(\mathbf{u}_i\mathbf{v}_i^T)\right\|_F$;
  $E_j \leftarrow R - \sum_{i=1}^j (\mathbf{s}_j)_i \cdot P_\Omega(\mathbf{u}_i\mathbf{v}_i^T)$;
  $U_j \leftarrow \begin{bmatrix} U_{j-1} & \mathbf{u}_j \end{bmatrix}$;
  $V_j \leftarrow \begin{bmatrix} V_{j-1} & \mathbf{v}_j \end{bmatrix}$;
end

waarbij $R_3 = U_3\,\mathrm{diag}(\mathbf{s}_3)V_3^T$ de rang-3 benadering is volgens algoritme 2. De gekende beoordelingen worden met algoritme 2 niet noodzakelijk exact geschat. Aangezien we ook geen begrenzingen opleggen aan de voorspellingen van het model $R_j = U_j\,\mathrm{diag}(\mathbf{s}_j)V_j^T$ is het mogelijk dat sommige voorspelde waarden buiten het interval $[0.01, 5]$ liggen. In de praktijk zouden we dus beter $L(R_j)$ als model gebruiken, waarbij $L(X)$ een operator is die bij invoer van een matrix $X \in \mathbb{R}^{m\times n}$ alle waarden begrensd tot het interval $[0.01, 5]$. *We gaan in deze opgave geen rekening houden met het begrenzen van de waarden tot het correcte interval.*

Om de nauwkeurigheid van de voorspellingen te evalueren gebruiken we de standaard *root mean square error* (RMSE) als maat. De RMSE tussen twee onvolledige matrices $A, B \in \mathbb{R}^{m\times n}$, voorgesteld als ijle matrices met eenzelfde ijlheidspatroon Ω en ζ niet-nul elementen, wordt gedefinieerd als

$$\mathrm{RMSE}(A, B) := \frac{1}{\sqrt{\zeta}}\|P_\Omega(A) - P_\Omega(B)\|_F.$$

De RMSE tussen R uit verg. (2) en $P_\Omega(R_3)$, waarbij Ω het ijlheidspatroon van R is, bedraagt dus

$$\frac{1}{\sqrt{22}}\left\|\begin{bmatrix} 0.0 & 0.0 & \mathbf{0.1} & \mathbf{0.0} & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \mathbf{-0.1} & 0.0 \\ \mathbf{0.0} & 0.0 & \mathbf{-0.1} & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \mathbf{0.0} & \mathbf{-0.1} & 0.0 \\ \mathbf{0.0} & \mathbf{0.0} & \mathbf{0.0} & \mathbf{0.0} & \mathbf{0.0} & \mathbf{0.0} & 0.0 & \mathbf{0.0} & \mathbf{0.1} & 0.0 & \mathbf{0.1} \\ 0.0 & 0.0 & \mathbf{0.0} & 0.0 & 0.0 & 0.0 & \mathbf{0.0} & 0.0 & 0.0 & 0.0 & \mathbf{-0.2} \\ \mathbf{0.0} & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \mathbf{0.1} & 0.0 \end{bmatrix}\right\|_F$$

ofwel ongeveer 0.0707.

Opdracht 10. Schrijf een functie met de hoofding

```
function [err] = SN_RMSE(A,B)
```

die bij invoer van twee ijle matrices $A, B \in \mathbb{R}^{m\times n}$ met eenzelfde ijlheidspatroon Ω als uitvoer de root mean square error als uitvoer `err` geeft. ♦

Opdracht 11. Zij `T` de matrix die je in opdracht 1 hebt ingeladen, Ω diens ijlheidspatroon, $\boldsymbol{\mu}$ de vector van de gemiddelde gekende beoordelingen van de gebruikers zoals bepaald door de functie `SN_userMeans` toegepast op de trainingdata `R` die je in opdracht 1 hebt ingeladen, en zij $\mathbf{1} \in \mathbb{R}^{2\,206}$ de vector waarin elk element 1 is. Hoeveel bedraagt de RMSE tussen `T` en de ijle matrix $P_\Omega(\mathbf{1}\boldsymbol{\mu}^T)$? ♦

Opdracht 12. Pas de Matlabfunctie `SN_rank1MatrixPursuit` toe op de gekende beoordelingenmatrix `R` die je hebt ingeladen in opdracht 1, waarbij je $r = 30$ als gewenste rang kiest en de testdata `T`, ingelezen in opdracht 1, als derde argument kiest. Maak een duidelijke figuur van de evolutie van de RMSE van algoritme 2, zoals deze gegeven wordt door het tweede uitvoerargument `rmse` van de functie `SN_rank1MatrixPursuit`. Neem de figuur op in het verslag. ♦

De modeloplossing berekent de gewenste figuur uit de voorgaande opgave in minder dan 5 minuten rekentijd op een Core i7-5600U (2.6GHz) met 8GB werkgeheugen zonder expliciet gebruik te maken van parallellisme.

Opdracht 13. Bereken `[U20,s20,V20,~] = SN_rank1MatrixPursuit(R,20,T)` en duid de factoren van de resulterende lagerangbenadering in de rest van het verslag aan met U_{20}, $\mathbf{s}_{20}$ en $\mathbf{V}_{20}$. Wat zijn de waarden van de vector $\mathbf{s}_{20}$, afgerond tot 1 cijfer na de komma? ♦

Opdracht 14. Schrijf een functie met de hoofding

```
function [movieIDs, score] = SN_actualBestMovies(R)
```

Het invoerargument is een ijle matrix $R \in \mathbb{R}^{m\times n}$ die de gekende gebruikersbeoordelingen bevat. De functie berekent de gemiddelde gebruikersbeoordeling voor elke film op basis van de gekende beoordelingen in R. Als uitvoer levert de functie een aflopend gesorteerde lijst `score` die deze gemiddelde beoordelingen bevat. De vector `movieIDs` bevat de permutatievector zodat `score(i)` de gemiddelde beoordeling van de kolom van R met volgnummer `movieIDs(i)` bevat. ♦

Opdracht 15. Schrijf een functie met de hoofding

```
function [movieIDs, score] = SN_predictedBestMovies(Uk,sk,Vk)
```

Zij $k \leq \min\{m,n\}$. Als invoer verwacht de functie de factoren $U_k \in \mathbb{R}^{m\times k}$, $\mathbf{s}_k \in \mathbb{R}^k$ en $V_k \in \mathbb{R}^{n\times k}$ van de lagerangontbinding $R_k = U_k \operatorname{diag}(\mathbf{s}_k) V_k^T$. De elementen van R_k worden verondersteld voorspelde gebruikersbeoordelingen voor te stellen. Het eerste uitvoerargument is een aflopend gesorteerde lijst `score` die de gemiddelde voorspelde beoordeling (over *alle* gebruikers) voor elke film bevat. De vector `movieIDs` bevat de permutatievector zodat `score(i)` de gemiddelde beoordeling van de kolom van R_k met volgnummer `movieIDs(i)` is.

De geheugencomplexiteit van je algoritme in functie van m, n en k mag hoogstens $\mathcal{O}(\max\{m,n\})$ bedragen. Hoe heb je deze doelstelling bereikt? ♦

Opdracht 16. Bereken met behulp van de functies uit de twee voorgaande opgaves de 25 films met de hoogste gemiddelde beoordelingen (op basis van de gekende beoordelingenmatrix `R`) en de 25 films met de hoogste gemiddelde voorspelde beoordelingen. Neem deze twee lijsten op in het verslag; geef de namen van de films, niet de volgnummers. Als model voor het voorspellen van de beoordelingen gebruik je het rang-20 model $(U_{20}, \mathbf{s}_{20}, V_{20})$ uit opdracht 13.

Welke lijst van films vind je het meest realistisch? Motiveer je keuze. Onderzoek in het bijzonder het aantal *gekende* beoordelingen in de trainingsdata `R` van de films in de twee lijsten en betrek deze informatie in je motivering. ♦

Natuurlijk kunnen we het aanbevelingssysteem gebruiken om gepersonaliseerde aanbevelingen te maken aan de gebruikers. De j^{de} kolom van het rang-k model $R_k := F_k W_k^T$ bevat namelijk de voorspelde beoordelingen voor de j^{de} gebruiker.

Opdracht 17. Schrijf een functie met de hoofding

```
function [movieIDs, score] = SN_predictedBestMoviesForUser(R,Uk,sk,Vk,j)
```

Zij $k \leq \min\{m,n\}$. Als eerste invoerargument verwacht de functie een onvolledige matrix $R \in \mathbb{R}^{m\times n}$ die de gekende beoordelingen bevat. De drie volgende argumenten zijn de factoren $U_k \in \mathbb{R}^{m\times k}$, $\mathbf{s}_k \in \mathbb{R}^k$ en $V_k \in \mathbb{R}^{n\times k}$ van de lagerangontbinding $R_k = U_k \operatorname{diag}(\mathbf{s}_k) V_k^T$. Het laatste invoerargument is het gebruikersnummer waarvoor we de beste films willen oplijsten. Het tweede uitvoerargument is een aflopend gesorteerde lijst `score` die de voorspelde beoordeling bevat voor de j^{de} gebruiker voor elke film *die de gebruiker nog niet heeft beoordeeld*, zoals bepaald door de gekende beoordelingen in R. De vector `movieIDs` bevat de permutatievector zodat `score(i)` de voorspelde beoordeling voor gebruiker j is van de film met volgnummer `movieIDs(i)`. Voor een gebruiker die bijvoorbeeld de films met volgnummers 9, 10 en 1987 reeds heeft beoordeeld, dient de kolomvector `movieIDs` dus een permutatie van alle filmnummers *behalve* 9, 10 en 1987 te bevatten. ♦

Opdracht 18. Gebruik het rang-20 model $R_{20} = U_{20} \operatorname{diag}(\mathbf{s}_{20}) V_{20}^T$ voor deze opdracht. Bereken de 10 beste films voor de gebruiker met volgnummer 98. Lijst de filmtitels op in het verslag samen met de voorspelde score voor gebruiker 98. Vergelijk deze lijst met de films die gebruiker 98 een score van minstens 4.5 gaf, *volgens de testdata* T. Neem ook deze lijst met filmtitels en bijhorende scores op in het verslag. Herhaal deze opdracht ook voor gebruiker 10100. Wat vind je van de aanbevelingen? ♦

4 Evaluatie

Opdracht 19. Hoeveel tijd heb je gespendeerd aan het oplossen van de opdrachten? Hoeveel tijd heb je gespendeerd aan het schrijven van het verslag? ⧫

Opdracht 20. In de loop van deze opgave hebben we allerhande veronderstellingen gemaakt om ons nieuw aanbevelingssysteem op te stellen. Wat zijn je bedenkingen hierbij? Vind je de resultaten realistisch? Zou je het ontwikkelde aanbevelingssysteem durven toevoegen aan de lijst van aanbevelingssystemen van MovieLens? ⧫

Opdracht 21. Welke bedenkingen heb je bij dit practicum? Was de opgave (veel) te gemakkelijk, (veel) te moeilijk of van een gepaste moeilijkheidsgraad? Wat zou je zelf anders aangepakt hebben? ⧫

A Code

```
function [U,s,V,rmse] = SN_rank1MatrixPursuit(R,k,T)
  [m, n] = size(R);
  rmse = zeros(k,1);
  U = zeros(m,k);
  V = zeros(n,k);
  U(:,1) = ones(m,1);
  V(:,1) = SN_userMeans(R);
  S = SN_sparseModel(U(:,1),1,V(:,1),R);
  P = SN_sparseModel(U(:,1),1,V(:,1),T);
  E = R - S;
  rmse(1) = SN_RMSE(T,P);
  for j = 2 : k
    [u,~,v] = svds(E,1);
    U(:,j) = u;
    V(:,j) = v;
    s = SN_optimalCoefficients(U(:,1:j),V(:,1:j),R);
    S = SN_sparseModel(U(:,1:j),s,V(:,1:j),R);
    P = SN_sparseModel(U(:,1:j),s,V(:,1:j),T);
    E = R - S;
    rmse(j) = SN_RMSE(T,P);
  end
end
```