Modellering en simulatie
Practicum 2: Lagerangbenaderingen

Wim Michiels en Nick Vannieuwenhoven

14 november 2016

1 Praktische informatie

Het practicum wordt individueel opgelost. Het verslag dient de volgende componenten te bevatten: de oplossingen van de opgaves en de broncode van je programma's. Het verslag wordt uiterlijk **vrijdag 23 december 2016** ingediend. Een papieren versie van het verslag dient zich vrijdagnacht 23 december 2016 om 23u59 in de studentenbrievenbus in gebouw 200A te bevinden. Deze uitdraai dient tevens de broncode van de programma's te bevatten. Daarenboven stuur je per elektronische post naar `nick.vannieuwenhoven@cs.kuleuven.be` een gecomprimeerde map met daarin een elektronische versie van het verslag alsook de Matlabbronbestanden. Zorg ervoor dat je bronbestanden de **opgelegde naamgeving** respecteren (zie verder) want de codes zullen automatisch geverifieerd worden. De namen van het verslag en de gecomprimeerde map (de bestandsextensie buiten beschouwing latende) dienen overeen te stemmen met je studentennummer; "s0123456.pdf" voor je verslag en "s0123456.zip," "s0123456.tar.gz" en dergelijke zijn allen aanvaardbaar.

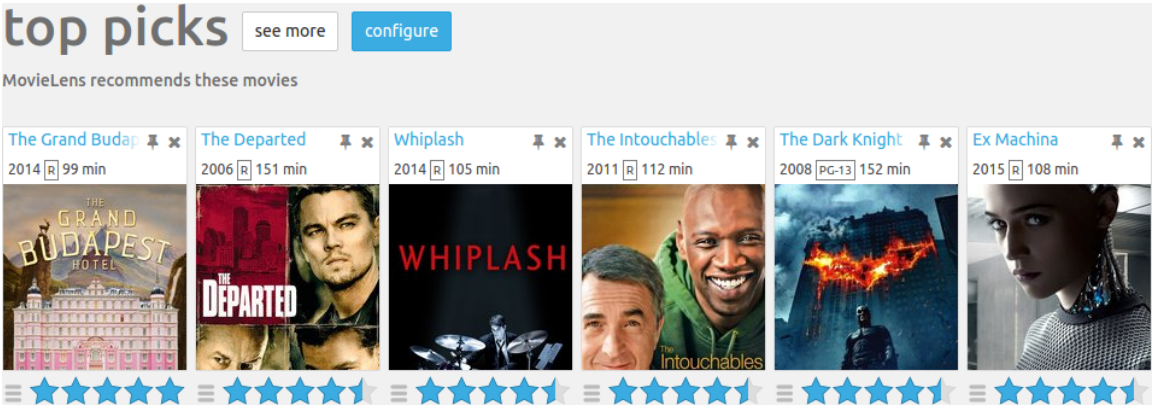
Veel succes!

> **In wat volgt vervang je elk voorkomen van "SN" met jouw studentennummer (r- of s-nummer)—bijvoorbeeld "s0123456". 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_".
>
> Het geniet de aanbeveling om de opgave eerst volledig te lezen alvorens met de opdrachten te starten.

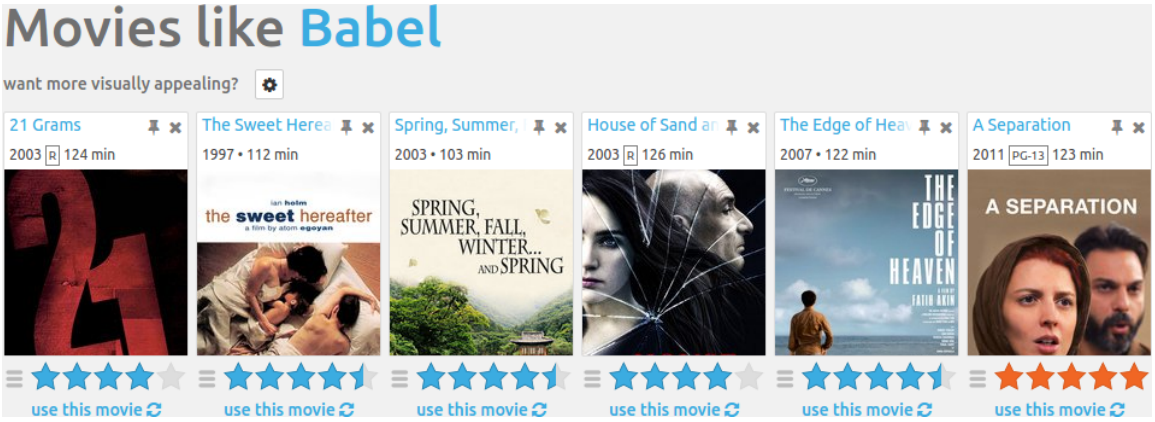
2 Een aanbevelingssysteem voor films

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 *Babel*. 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



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



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

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 januari 2016. De volledige databank bevat ongeveer 22 miljoen beoordelingen van zo'n 240 000 geregistreerde gebruikers voor circa 33 000 films. Teneinde de geheugenvereisten voor dit practicum te beperken, zullen we met circa 20% van de volledige dataset werken. Deze beperktere dataset bevat 4 901 films en 48 769 gebruikers die tezamen 4 231 228 beoordelingen produceerden. De deelverzameling werd opgesteld door eerst de films te reduceren tot films die ten vroegste op 1 januari 2000 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.

2.1 Matrixvervolledigiing

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 *The Dark Knight* (2008), *Wall·E* (2008), *Batman Begins* (2005), *Pan's Labyrinth* (2006), *Up* (2009) en *Ratatouille* (2007). De beoordelingen van 12 willekeurige gebruikers uit MovieLens' databank zou men dan als volgt kunnen voorstellen:

$$R^T = \begin{array}{r} \text{The Dark Knight} \\ \text{Wall·E} \\ \text{Batman Begins} \\ \text{Pan's Labyrinth} \\ \text{Up} \\ \text{Ratatouille} \end{array} \begin{bmatrix} \mathbf{4.0} & \mathbf{4.0} & \mathbf{4.5} & ? & \mathbf{3.0} & ? & ? & \mathbf{4.0} & ? & \mathbf{4.0} & ? & \mathbf{4.0} \\ \mathbf{4.5} & ? & \mathbf{3.5} & \mathbf{4.0} & ? & \mathbf{3.5} & \mathbf{5.0} & \mathbf{4.5} & \mathbf{5.0} & ? & \mathbf{4.5} & \mathbf{3.0} \\ ? & \mathbf{3.0} & \mathbf{4.0} & \mathbf{3.5} & \mathbf{2.5} & \mathbf{3.0} & \mathbf{4.0} & ? & \mathbf{5.0} & \mathbf{4.5} & ? & ? \\ \mathbf{5.0} & ? & ? & ? & ? & \mathbf{4.0} & ? & \mathbf{4.0} & \mathbf{4.5} & \mathbf{5.0} & \mathbf{5.0} & ? \\ ? & \mathbf{5.0} & ? & \mathbf{3.5} & \mathbf{4.0} & ? & \mathbf{3.5} & \mathbf{4.0} & ? & ? & \mathbf{3.5} & \mathbf{2.0} \\ \mathbf{3.5} & \mathbf{3.0} & \mathbf{4.0} & \mathbf{3.0} & \mathbf{3.5} & \mathbf{3.5} & \mathbf{4.0} & ? & \mathbf{3.5} & \mathbf{1.5} & \mathbf{4.0} & \mathbf{2.0} \end{bmatrix}. \quad (1)$$

Elke rij van R geeft de beoordeling van één gebruiker voor enkele van de films en elke kolom 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 voorspellen van de i^{de} rij van de matrix R. Duid het aantal gebruikers van het aanbevelingsysteem aan met m en het aantal films uit de catalogus 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

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

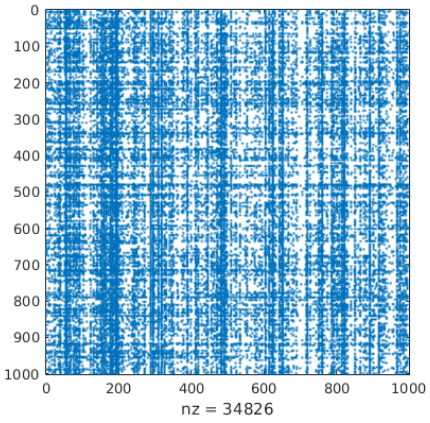
hetgeen equivalent is met

$$R \approx WF^T = \mathbf{w}_1\mathbf{f}_1^T + \mathbf{w}_2\mathbf{f}_2^T + \cdots + \mathbf{w}_r\mathbf{f}_r^T \tag{2}$$

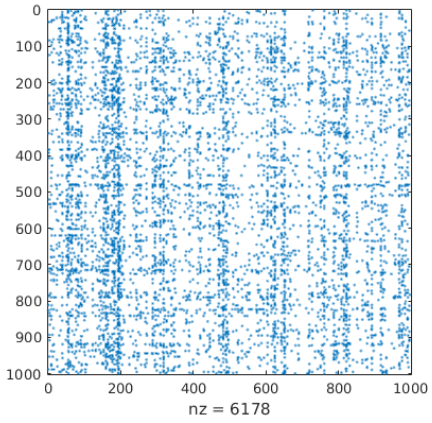
en waarbij $\mathbf{w}_i \in \mathbb{R}^m$, respectievelijk $\mathbf{f}_i \in \mathbb{R}^n$, de i^{de} kolom is van de matrix $W \in \mathbb{R}^{m \times r}$, respectievelijk $F \in \mathbb{R}^{n \times r}$. We kunnen model (2) als volgt interpreteren. Elke vector $\mathbf{f}_i$ kunnen we interpreteren 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 Dark Knight* en de animatiefilm *Up*. 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 $\mathbf{w}_1\mathbf{f}_1^T$ bevat dan op positie (i, j) de bijdrage van het kenmerk "geweld" aan de totaalscore van gebruiker i voor film j.

Opdracht 1. Laad het bestand `MovieLens20M_Subset.mat` in. Dit bestand bevat de volgende elementen: `movieLabel` is een *cell array* die de titels bevat van de 4 901 films in de beperkte catalogus; `R` is de 48 769 × 4 901 onvolledige beoordelingenmatrix die 3 596 544 gekende beoordelingen bevat, compact voorgesteld als een ijle matrix; en `T` is eveneens een 48 769 × 4 901 onvolledige beoordelingenmatrix die 634 684 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 gebruiker met volgnummer i. De j^{de} kolom van `R` en `T` bevat de gekende beoordelingen van de film met titel `movieLabel{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 WF^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 $R \approx WF^T$ 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 $m \geq n$ van rang $r \leq n$ gegeven wordt door de *afgeknotte singuliereswaardenontbinding*. Laat de singuliereswaardenontbinding van X gegeven zijn door

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

waarbij $U \in \mathbb{R}^{m\times m}$, $\Sigma \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{n\times n}$, $U_1 \in \mathbb{R}^{m\times r}$, $\Sigma_1 \in \mathbb{R}^{r\times r}$ 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 formeel 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.

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 $r \leq \min\{m, n\}$. Je mag veronderstellen dat de singuliere waardenontbinding van A uniek is. Onderstel dat algoritme 1 op A wordt toegepast. Bewijs dat σ_j de j^{de} grootste singuliere waarde van A is en dat

$$A - E_k = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{v}_j^T$$

de rang-k (met $k \leq r$) afgeknotte singuliere waardenontbinding van A is. Leidt hieruit af dat algoritme 1 na exact r stappen stopt. ♦

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

Opdracht 4. Stel dat we de stappen 4–6 van algoritme 1 slechts éénmaal zouden uitvoeren en veronderstel dat het berekenen van de beste rang-1 benadering van een $m\times n$ matrix slechts $m+n$ dubbele-precisie vlottendekommagetallen (van 8 byte) vereist (om het resultaat voor te stellen). Kan men dit algoritme dan toepassen op een $240\,000 \times 33\,000$ matrix met 22 miljoen niet-nulwaarden (i.e., de volledige MovieLens databank) op een laptop met 8GB werkgeheugen en 32GB swap geheugen? Waarom wel of niet? ♦

Om dit probleem op te lossen zal het algoritme van Wang et al. er voor 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

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

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

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

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

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:

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

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

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:

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

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:

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

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. 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? ♦

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

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 $r_{i,j} \leftarrow r_{i,j} - \mu_i$, waarbij μ_i het gemiddelde is van alle gekende beoordelingen van gebruiker i. Dit heeft als effect dat de individuele verschillen in de beoordelingsstijl van de gebruikers getemperd worden.

Opdracht 7. Schrijf een functie met de hoofding

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

die als uitvoer een kolomvector `mu` van lengte m 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 8. 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 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.

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

$E_0 \leftarrow R$;
$\mathbf{u}_1 \leftarrow$ `SN_userMeans(R)`;
$\mathbf{v}_1 \leftarrow \mathbf{1}$;
$E_1 \leftarrow R - P_\Omega(\mathbf{u}_1\mathbf{v}_1)$;
$U_1 \leftarrow \mathbf{u}_j$;
$V_1 \leftarrow \mathbf{v}_j$;
$\mathbf{s}_1 \leftarrow [1]$;
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 $\|E_{j-1} - \sigma_j \mathbf{u}_j \mathbf{v}_j^T\|_F = \min_{\operatorname{rank}(B)\leq 1} \|E_{j-1} - B\|_F$;
  Zij $\mathbf{s}_j \in \mathbb{R}^j$ zodanig dat $\left\|R - \sum_{i=1}^j (\mathbf{s}_j)_i \cdot P_\Omega(\mathbf{u}_i\mathbf{v}_i^T)\right\|_F = \min_{\mathbf{x}\in\mathbb{R}^j} \left\|R - \sum_{i=1}^j x_i \cdot P_\Omega(\mathbf{u}_i\mathbf{v}_i^T)\right\|_F$;
  $E_j \leftarrow R - \sum_{i=1}^j (\mathbf{s}_j)_i \cdot P_\Omega(\mathbf{u}_i\mathbf{v}_i^T)$;
  $U_j \leftarrow \begin{bmatrix} U_{j-1} & \mathbf{u}_j \end{bmatrix}$;
  $V_j \leftarrow \begin{bmatrix} V_{j-1} & \mathbf{v}_j \end{bmatrix}$;
end

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:

$$R^T \approx R_3^T = \begin{array}{r} \\ \text{The Dark Knight} \\ \text{Wall·E} \\ \text{Batman Begins} \\ \text{Pan's Labyrinth} \\ \text{Up} \\ \text{Ratatouille} \end{array} \begin{bmatrix} \mathbf{4.5} & \mathbf{3.6} & \mathbf{4.0} & 3.6 & \mathbf{3.1} & 3.5 & 4.3 & \mathbf{4.2} & 4.8 & \mathbf{4.3} & 4.5 & \mathbf{3.1} \\ \mathbf{4.5} & 3.6 & \mathbf{4.0} & \mathbf{3.6} & 3.1 & \mathbf{3.5} & \mathbf{4.3} & \mathbf{4.2} & \mathbf{4.7} & 4.2 & \mathbf{4.5} & \mathbf{3.1} \\ 4.5 & \mathbf{3.2} & \mathbf{4.0} & \mathbf{3.6} & \mathbf{2.7} & \mathbf{3.5} & \mathbf{4.5} & 4.2 & \mathbf{4.9} & \mathbf{4.5} & 4.8 & 3.4 \\ \mathbf{4.7} & 3.9 & 4.1 & 3.8 & 3.0 & \mathbf{3.6} & 4.3 & \mathbf{4.2} & \mathbf{5.0} & \mathbf{5.0} & \mathbf{4.6} & 3.2 \\ 4.2 & \mathbf{5.1} & 4.1 & \mathbf{3.5} & \mathbf{4.2} & 3.7 & \mathbf{3.6} & \mathbf{4.0} & 4.3 & 3.4 & \mathbf{3.6} & \mathbf{2.0} \\ \mathbf{3.4} & \mathbf{3.0} & \mathbf{4.0} & \mathbf{3.1} & \mathbf{3.5} & \mathbf{3.4} & \mathbf{4.1} & 4.2 & \mathbf{3.6} & \mathbf{1.4} & \mathbf{3.9} & \mathbf{2.1} \end{bmatrix},$$

waarbij $R_3 = U_3 \operatorname{diag}(\mathbf{s}_3) V_3^T$ 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 $R_j = U_j \operatorname{diag}(\mathbf{s}_j) V_j^T$ 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

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

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

$$\frac{1}{\sqrt{48}} \left\| \begin{bmatrix} \mathbf{-0.5} & \mathbf{0.4} & \mathbf{0.5} & 0.0 & \mathbf{-0.1} & 0.0 & 0.0 & \mathbf{-0.2} & 0.0 & \mathbf{-0.3} & 0.0 & \mathbf{0.9} \\ \mathbf{0.0} & 0.0 & \mathbf{-0.5} & \mathbf{0.4} & 0.0 & \mathbf{0.0} & \mathbf{0.7} & \mathbf{0.3} & \mathbf{0.3} & 0.0 & \mathbf{0.0} & \mathbf{-0.1} \\ 0.0 & \mathbf{-0.2} & \mathbf{0.0} & \mathbf{-0.1} & \mathbf{-0.2} & \mathbf{-0.5} & \mathbf{-0.5} & 0.0 & \mathbf{0.1} & \mathbf{0.0} & 0.0 & 0.0 \\ \mathbf{0.3} & 0.0 & 0.0 & 0.0 & 0.0 & \mathbf{0.4} & 0.0 & \mathbf{-0.2} & \mathbf{-0.5} & \mathbf{0.0} & \mathbf{0.4} & 0.0 \\ 0.0 & \mathbf{-0.1} & 0.0 & \mathbf{0.0} & \mathbf{-0.2} & 0.0 & \mathbf{-0.1} & \mathbf{0.0} & 0.0 & 0.0 & \mathbf{-0.1} & \mathbf{0.0} \\ \mathbf{0.1} & \mathbf{0.0} & \mathbf{0.0} & \mathbf{-0.1} & \mathbf{0.0} & \mathbf{0.1} & \mathbf{-0.1} & 0.0 & \mathbf{-0.1} & \mathbf{0.1} & \mathbf{0.1} & \mathbf{-0.1} \end{bmatrix} \right\|_F$$

oftewel ongeveer 0.2940.

Opdracht 9. 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 10. 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 $\mathbf{1} \in \mathbb{R}^{4\,901}$ de vector waarin elk element 1 is. Hoeveel bedraagt de RMSE tussen `T` en de ijle matrix $P_\Omega(\boldsymbol{\mu}\mathbf{1}^T)$? ♦

Opdracht 11. 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 12. 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 2 cijfers na de komma? ♦

Opdracht 13. 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 14. 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 (van *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 15. 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 12.

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. ♦

2.2 Clustering

Om gelijkaardige films te detecteren zullen we de beoordeling van film i te modelleren als een discrete stochastische variabele F_i die de waarden $\{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$ kan aannemen. Een maat voor de statistische afhankelijkheid tussen de twee stochastische variabelen F_i en F_j is de zogenaamde *Pearson correlatiecoefficiënt*. Deze correlatiecoefficiënt is een reëel getal tussen -1 en 1. Een waarde dicht bij 1 geeft aan dat de stochastische variabelen F_i en F_j sterk positief verbonden zijn, een waarde dicht bij -1 duidt op een sterk negatief verband en een waarde rond 0 impliceert dat er geen bijzonder lineair verband bestaat tussen de beoordelingen van film i en j. In dit practicum zullen we een film i als gelijkaardig beschouwen aan een film j wanneer hun beoordelingen onderling sterk positief gecorreleerd zijn.

In de vorige sectie werd een methode besproken om de ontbrekende waarden in de onvolledige beoordelingenmatrix R te vervangen door zinvolle waarden, resulterende in de voorspelde beoordelingenmatrix, bijvoorbeeld de rang-k ontbinding $R_k = U_k \operatorname{diag}(\mathbf{s}_k)V_k^T \approx R$. Elk van de $n = 4\,901$ kolommen van R_k bevat nu de (voorspelde) beoordelingen voor elk van de $m = 48\,769$ gebruikers. We kunnen nu de correlatiematrix $C \in \mathbb{R}^{n\times n}$ van de stochastische variabelen $F_1, F_2, \ldots, F_n$ schatten op basis van de (voorspelde) steekproef R_k. Deze steekproefcorrelatiematrix wordt als volgt elementsgewijs gedefinieerd:

$$c_{i,j} = \operatorname{corr}(F_i, F_j) = \frac{1}{m-1}\frac{\sum_{k=1}^{m}(a_{k,i} - \nu_{F_i})(a_{k,j} - \nu_{F_j})}{\sigma_{F_i}\sigma_{F_j}},$$

waarbij

$$\nu_{F_i} = \frac{1}{m}\sum_{k=1}^{m} a_{k,i} \quad \text{en} \quad \sigma_{F_i} = \sqrt{\frac{1}{m-1}\sum_{k=1}^{m}(a_{k,i} - \nu_{F_i})^2}$$

respectievelijk het steekproefgemiddelde en de steekproefstandaardafwijking zijn. Zij

$$\Sigma = \operatorname{diag}(\sigma_{F_1}, \sigma_{F_2}, \ldots, \sigma_{F_n})$$

een diagonaalmatrix die de steekproefstandaardafwijkingen bevat van de stochastische variabelen F_i, $\boldsymbol{\nu}^T = \begin{bmatrix}\nu_{F_1} & \nu_{F_2} & \cdots & \nu_{F_n}\end{bmatrix} \in \mathbb{R}^n$ een vector die de steekproefgemiddelden van de statistische variabelen F_i bevat en $\mathbf{1}^T = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix} \in \mathbb{R}^m$. De steekproefcorrelatiematrix wordt dan expliciet gegeven door

$$C = \frac{1}{m-1}\big((R_k - \mathbf{1}\boldsymbol{\nu}^T)\Sigma^{-1}\big)^T\big((R_k - \mathbf{1}\boldsymbol{\nu}^T)\Sigma^{-1}\big). \tag{8}$$

Zo wordt de steekproefcorrelatiematrix van de voorspelde beoordelingenmatrix R_3 in het voorbeeld uit de voorgaande sectie afgerond tot 2 cijfers na de komma gegeven door

$$C = \begin{bmatrix} 1.00 & 1.00 & 0.96 & 0.95 & 0.29 & 0.31 \\ 1.00 & 1.00 & 0.96 & 0.95 & 0.30 & 0.34 \\ 0.96 & 0.96 & 1.00 & 0.92 & 0.01 & 0.21 \\ 0.95 & 0.95 & 0.92 & 1.00 & 0.24 & 0.02 \\ 0.29 & 0.30 & 0.01 & 0.24 & 1.00 & 0.44 \\ 0.31 & 0.34 & 0.21 & 0.02 & 0.44 & 1.00 \end{bmatrix}.$$

Opdracht 16. Schrijf een functie met hoofding

```
function [C] = SN_correlationMatrix(Uk,sk,Vk)
```

die de steekproefcorrelatiematrix $C \in \mathbb{R}^{n \times n}$ berekent van de steekproef in de $m \times n$ matrix $U_k \operatorname{diag}(\mathbf{s}_k) V_k^T$, waarbij $U_k \in \mathbb{R}^{m \times k}$, $\mathbf{s}_k \in \mathbb{R}^k$ en $V_k \in \mathbb{R}^{n \times k}$.

De geheugencomplexiteit van je algoritme in functie van m, n en k mag hoogstens $\mathcal{O}(k^2+n^2)$ bedragen. Hoe heb je deze doelstelling bereikt? ♦

Opdracht 17. Schrijf een functie met hoofding

```
function [moviedIDs] = SN_similarMovies(C,m,n)
```

die op basis van een $4\,901 \times 4\,901$ correlatiematrix C (zoals deze uit de voorgaande opgave) de `n` volgnummers van de films die het sterkst positief gecorreleerd zijn met de film met volgnummer `m` berekent. Het uitvoerargument is een vector van lengte `n` die de volgnummers bevat. Het eerste element in deze vector is steeds `m`. ♦

Opdracht 18. Welke n films zijn het sterkst positief gecorreleerd met de film met volgnummer k, voor de volgende waarden van n en k? Gebruik de correlatiematrix die je met de functie uit opdracht 16 berekent op basis van het rang-20 model $(U_{20}, \mathbf{s}_{20}, V_{20})$.

n	k	titel
10	446	*Harry Potter and the Chamber of Secrets* (2002)
5	3854	*Captain America: The First Avenger* (2011)
3	4355	*Zero Dark Thirty* (2012)
3	2747	*Futurama: Bender's Game* (2008)
9	160	*Monsters, Inc.* (2001)

Zijn deze resultaten realistisch? Motiveer. Zijn er gelijkenissen met de aanbevelingen van andere aanbevelingssystemen, zoals bijvoorbeeld IMDB.com, MovieLens, Amazon.com of Netflix? ♦

Opdracht 19. Bewijs dat de correlatiematrix C in vergelijking (8) hoogstens rang $r + 1$ heeft wanneer de voorspelde beoordelingenmatrix A exact rang r heeft. Hint: denk aan verg. (2). ♦

3 Evaluatie

Opdracht 20. Hoeveel tijd heb je gespendeerd aan het oplossen van de opdrachten? Hoeveel tijd heb je gespendeerd aan het schrijven van het verslag? ♦

Opdracht 21. 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 22. 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) = SN_userMeans(R);
  V(:,1) = ones(n,1);
  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
```