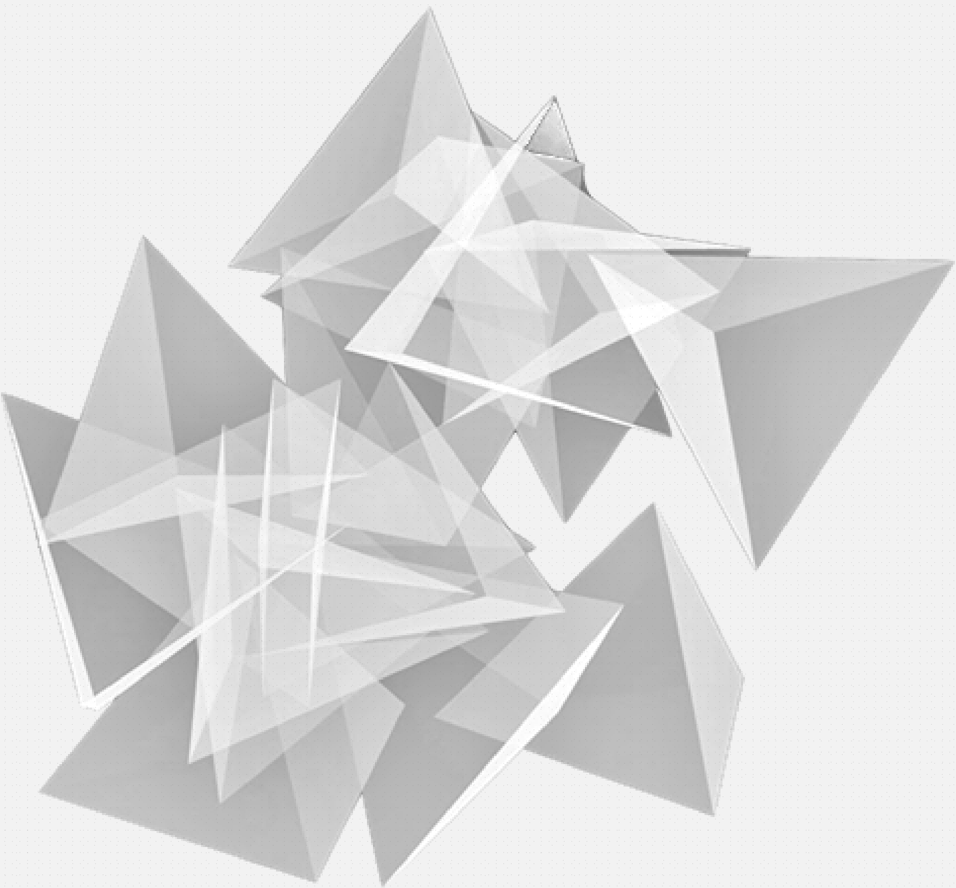
Practicum 1 - Lagerangbenaderingen

Bruno Vandekerkhove



Academisch Jaar 2019

G0Q57A: Modellering & Simulatie

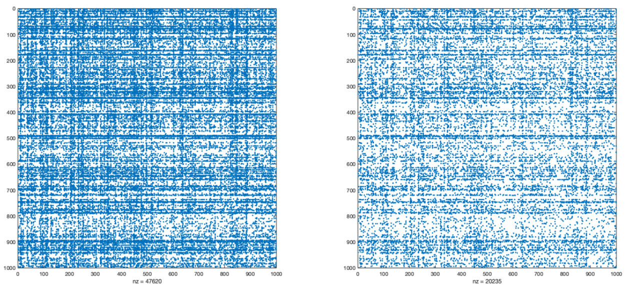
Inhoudsopgave

Een Aanbevelingssysteem voor Films

De broncode bevindt zich in de `src` folder. Het algemene script (`src/s0216676_script`) is opgedeeld in secties, één per opgave. De afzonderlijke opgaven worden hieronder beantwoord. Aan het einde van elk antwoord wordt (indien nodig) de broncode weergegeven.

Opdracht 1

We laden de dataset in met de `load` functie. De uitvoer staat weergegeven in figuur 1.



Figuur 1: Grafische voorstelling van ijle matrices R en T.

```
set(0, 'defaultFigurePosition', get(0, 'Screensize')); % Figuren vullen scherm
load('MovieLens_Subset.mat');
subplot(1,2,1)
spy(R(1:1000,1:1000))
subplot(1,2,2)
spy(T(1:1000,1:1000))
```

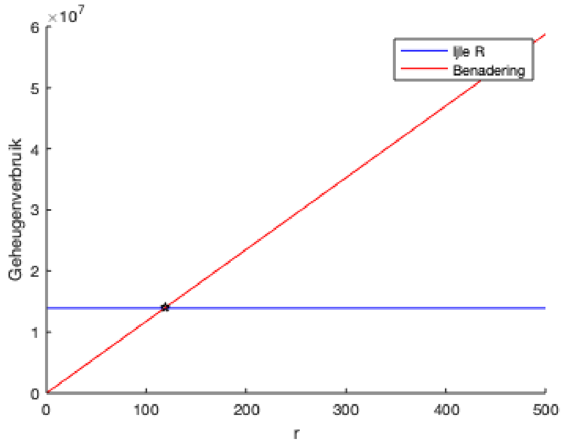
Opdracht 2

Stel dat gehele getallen 4 - en reële getallen 8 bytes innemen. De volle matrix `full(R)` zou dan 220388224 bytes (≈ 210MB) in beslag nemen ; 1 reëel getal per element. Voor de ijle matrix `sparse(R)` lijkt het op het eerste gezicht 18525728 bytes (< 18MB) te zijn ; 2 gehele getallen en 1 reëel getal per element dat niet nul is. MatLab verbruikt echter iets meer dan dat. Als men de juiste formule toepast voor het geheugenverbruik van ijle matrices [1] :

$$12 \times nnz + 4 \times n$$

[1] *`MATLAB` gebruikt het `CSC` formaat voor ijle matrices : "Even though `MATLAB` is written in `C`, it follows its `LINPACK` and `Fortran` predecessors and stores full matrices by columns. This organization has been carried over to sparse matrices. A sparse matrix is stored as the concatenation of the sparse vectors representing its columns. Each sparse vector consists of a floating point array of*

dan komt men uit op bijna 14 miljoen bytes. Op mijn computer was het totaal meer dan 18,6 miljoen omdat gehele getallen daar met 8 bytes worden voorgesteld.



Figuur 2: Geheugenverbruik voor ijle matrix R en lagerangsbenadering. Het snijpunt bevindt zich in $r \approx 119$.

```
[m,n] = size(R);
ratings = nnz(R);
int_mem = 4;
double_mem = 8;
max_r = 500;
%
fprintf('Geheugenruimte full(R) : %i\n', m * n * double_mem)
%
size_sparse_naive = ratings * (int_mem * 2 + double_mem);
size_sparse = 12 * ratings + 4 * n;
fprintf('Geheugenruimte sparse(R) : %i\n', size_sparse)
%
fullR = full(R);
fprintf('Matlab zelf gebruikt respectievelijk %i en %i bytes.\n', whos('fullR').bytes, ...
    whos('R').bytes)
%
r = 1:max_r;
size_approx = (m + n) * double_mem * r;
snijpunt_r = size_sparse / ((m + n) * double_mem);
fprintf('Snijpunt in r = %i\n\n', snijpunt_r)
%
hold on
plot(r, repmat(size_sparse,1,max_r), 'b-')
plot(r, size_approx, 'r-')
plot(snijpunt_r, size_sparse, 'kp')
xlabel('r')
ylabel('Geheugenverbruik')
legend('Ijle R', 'Benadering', 'Location', 'northeast')
```

Opdracht 3

Men weet dat

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{1}$$

Het iteratief algoritme gaat bij elke stap σ_j, u_j en v_j bepalen zodat de Frobeniusnorm van $E_{j-1} - \sigma_j u_j v_j^T$ (de nieuwe E_j) minimaal is. Dit komt overeen met het bepalen van een afgeknotte singulierewaardenontbinding van

nonzero entries (or two such arrays for complex matrices), together with an integer array of row indices. A second integer array gives the locations in the other arrays of the first element in each column. Consequently, the storage requirement for an $m \times n$ real sparse matrix with nnz nonzero entries is nnz reals and $nnz + n$ integers. On typical machines with 8-byte reals and 4-byte integers, this is $12nnz + 4n$ bytes." [1]

graad 1. De σ_j dient daarbij de grootste singuliere waarde van E_{j-1} te zijn. Stel $j = 0$, dan :

$$E_1 = E_0 - \sigma_j u_j v_j^T = A - \sigma_j u_j v_j^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T - \sigma_j u_j v_j^T = \sum_{i=2}^{r} \sigma_i u_i v_i^T$$

Na een aantal iteraties bekomt men uiteindelijk :

$$E_k = \sum_{i=k+1}^{r} \sigma_i u_i v_i^T$$

Want telkens wordt er een afgeknotte singulierewaardenontbinding bepaald van rang 1 zodat de term horende bij de grootste singuliere waarde keer op keer verdwijnt na aftrekking, totdat de laatste $r - k$ termen overblijven. Men kan opmerken dat dit niet verreist dat de u_j en v_j bekomen in stap 4 overeenstemmen met de u_i en v_i in (1). Het teken van u_j en v_j kan bijvoorbeeld verschillen.

Berekent men ten slotte $A - E_k$ dan bekomt men het gevraagde :

$$A - E_k = \sum_{i=1}^{r} \sigma_i u_i v_i^T - \sum_{i=k+1}^{r} \sigma_i u_i v_i^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

Opdracht 4

Gezien mijn studentennummer $s0216676$ is, is c gelijk aan 6. Met $sizeof(double) = 8$ zal stap 4 een totaal aan $8 * (m + n + 1)$ bytes nodig hebben. Voor stap 5 krijgt men te maken met de matrix E die maar één keer wordt gealloceerd en $8 * n * m$ bytes nodig heeft. Stap 6 verbruikt 4 bytes (voor het geheel getal). Het aantal keren dat men de stappen uitvoert maakt niet uit.

Men heeft dus te maken met $((280000 + 58000 + 1) * 8 + (280000 * 58000 * 8) + 4)/1024^3 \approx 120$ GB wat op zich al meer is dan het optimistische $8 + 2 + (c + 1)^2 = 8 + 2 + 49 = 59$ GB. Als men met ijle voorstellingen blijft werken kan hier iets aan gedaan worden.

Opdracht 5

De drie vectoren `[i,j,x]` die teruggegeven worden door `find(A)` verbruiken elk $\mathcal{O}(\zeta)$ geheugen. In de implementatie van de functie `s0216676_sparseModel` wordt `[x]` zélf hergebruikt om de resultaten van de lineaire operator $P_\Omega(X)$ op te slaan. Elk element x_{ij} is gelijk aan de vermenigvuldiging van rij i van de matrix $Uk * diag(sk)$ en kolom j van Vk^T. Het volstaat dus elementsgewijs te vermenigvuldigen van rij i van Uk met sk^T en dit te vermenigvuldigen met de getransponeerde rij j van Vk.

```
function [P] = s0216676_sparseModel(Uk,sk,Vk,A)
    [i,j,x] = find(A);
    for idx = 1:nnz(A)
        x(idx) = Uk(i(idx),:) .* sk' * Vk(j(idx),:)';
    end
    P = sparse(i, j, x);
end
```

Opdracht 6

Opdracht 7

Opdracht 8

Referenties

[1] John R. Gilbert, Cleve Moler, and Robert Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356, January 1992.

Evaluatie