

MAT1120 Oblig 2

Daniel Heinesen, daniehei

31. oktober 2016

Oppgave 1 Vi har en $m \times 1$ vektor \mathbf{u} og en $1 \times n$ vektor \mathbf{v}^T . Ytterproduktet mellom disse 2 er gitt ved:

$$\begin{aligned}\mathbf{u}\mathbf{v}^T &= \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} [v_1 \dots v_n] \\ &= \begin{bmatrix} u_1 v_1 & \dots & u_1 v_n \\ \vdots & \ddots & \vdots \\ u_m v_1 & \dots & u_m v_n \end{bmatrix} = [v_1 \mathbf{u} \dots v_n \mathbf{u}]\end{aligned}$$

Siden v_i er komponenter og skalarer ser vi at alle kolonnene er lineært avhengige og

$$\mathbf{u}\mathbf{v}^T = \text{span}\{\mathbf{u}\} \quad (1)$$

Rang er definert som $\dim(\text{col}(\mathbf{u}\mathbf{v}^T))$. Vi kan se at

$$\text{col}(\mathbf{u}\mathbf{v}^T) = \mathbf{u}$$

og

$$\text{rank}(\mathbf{u}) = \dim(\text{col}(\mathbf{u}\mathbf{v}^T)) = 1 \quad (2)$$

```
>> u = rand(4,1);
>> v = rand(5,1);
>> u*v'
ans =

    0.488594    0.164829    0.504212    0.729964    0.669458
    0.043800    0.014776    0.045200    0.065438    0.060014
    0.327409    0.110453    0.337874    0.489151    0.448606
    0.227790    0.076846    0.235071    0.340320    0.312111

>> rank(u*v')
ans = 1
```

Oppgave 2 Vi skal vise at:

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

Fra oppgaveteksten får vi at

$$\mathbf{A} = \mathbf{B}\mathbf{C} = \sum_{j=1}^n \text{Col}_j(\mathbf{B}) \text{Row}_j(\mathbf{C})$$

Vi setter at $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}$ og $\mathbf{C} = \mathbf{V}^T$. Vi har da

$$\mathbf{A} = \mathbf{B}\mathbf{C} = (\mathbf{U}\mathbf{\Sigma})(\mathbf{V}^T) = \sum_{j=1}^n \text{Col}_j(\mathbf{U}\mathbf{\Sigma}) \text{Row}_j(\mathbf{V}^T)$$

Siden $\mathbf{\Sigma}$ er en diagonalmatrise vil

$$\begin{aligned}\mathbf{U}\mathbf{\Sigma} &= \begin{bmatrix} \sigma_1 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \cdots & \ddots & \sigma_r & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_m \end{bmatrix} \\ &= \begin{bmatrix} \sigma_1 \mathbf{u}_1 & \cdots & \sigma_r \mathbf{u}_r & \cdots & 0 \end{bmatrix}\end{aligned}$$

og får da

$$Col_j(\mathbf{U}\mathbf{\Sigma}) = \sigma_j \mathbf{u}_j$$

For $Row_j(\mathbf{V}^T)$:

$$\mathbf{V}^T = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}^T = \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_n^T \end{bmatrix}$$

og da blir

$$Row_j(\mathbf{V}^T) = \mathbf{v}_j^T$$

Vi får da til slutt at

$$\mathbf{A} = \sum_{j=1}^n Col_j(\mathbf{U}\mathbf{\Sigma}) Row_j(\mathbf{V}^T) = \sum_{j=1}^n \sigma_j \mathbf{u}_j \mathbf{v}_j^T \quad (3)$$

Oppgave 3

a)

*Kommentar/Advarsel: All koden i denne obligen er skrevet i **Octave**. Jeg håper at det også skal fungere i Matlab, jeg har ikke hatt muligheten til å teste dette.*

```
function AK=svdApprox(A,k)

A = double(A);

if k>rank(A)
    disp("The rank of A is lower than k");
    return
end

[U,S,V] = svd(A);
AK = zeros(size(A));

for j=1:(k)
    AK = AK + S(j,j)*U(:,j)*V(:,j)';
end

AK = uint8(AK); #If AK is double the image isnt saved correctly
```

b) Vi laster inn bildet som en matrise og sjekker rangen den med **rank**-kommandoen og **rank**(A, ϵ) med $\epsilon = 0.001$

```
>> A = imread("mm.gif","gif");  
>> A = double(A);  
>> rank(A)  
ans = 256  
>> rank(A,0.001)  
ans = 256
```

Vi kan se at begge kommandoene gir samme svar $\text{rank}(A) = 256$.

c) For $k = 8$



Figur 1: mm.gif svd-dekomponert med $k = 8$

og for $k = 32$:



Figur 2: mm.gif svd-dekomponert med $k = 32$

Med $k = 8$ kan vi såvidt se at hvem bildet er av, og for $K = 32$ er kan man helt tydelig se motivet. Man kan likevel se at bildet har blitt komprimert, særlig rundt halsen og håret. Vi burde med andre ord velge en noe større k for at bildet skal ha en god kvalitet.

d) Figur 1 i oppgaveteksten består av nøyanser av sort til grå. Gjør vi dette til en matrise kan vil kolonnene bestå av de forskjellige fargene". Alle elementene i en kolonne er like, hvilket betyr at radene i denne matrisen er like. Siden vi kan skrive rangen til denne matrisen som

$$\text{Rank}(\mathbf{A}) = \dim(\text{Col}(\mathbf{A})) = \dim(\text{Row}(\mathbf{A}))$$

og alle radene er like, betyr det at dimensjonen på radrommet er 1, og derfor er rangen til dette bildet 1.

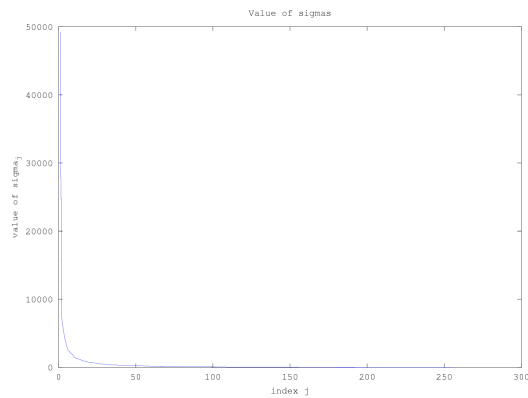
e) Jeg laget et lite program jeg laget for å plotte σ_j

```
function plotSigma(A,k)
    if k>rank(A)
        disp("The rank of A is lower than k");
        return
    end
    A = double(A);
    [U,S,V] = svd(A);

    s = diag(S);

    graphics_toolkit("gnuplot")
    figure(1);
    plot(s(1:k));
    title("Value of sigmas");
    xlabel("index j");
    ylabel("value of sigma_j");
    saveas(1,"sigmas.png");
```

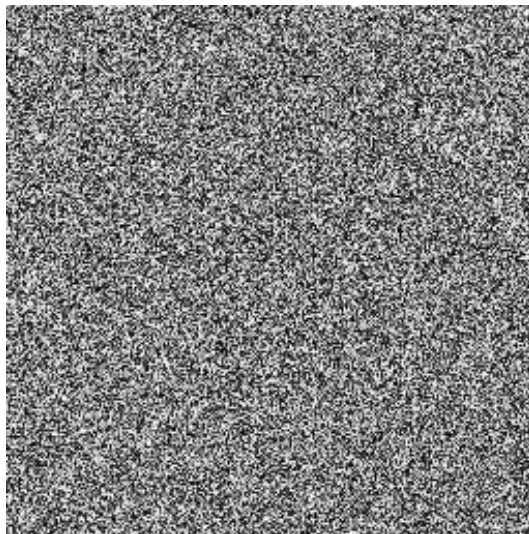
Plotter vi alle sigmaene får vi:



Figur 3: Sigmaverdiene for mm.gif

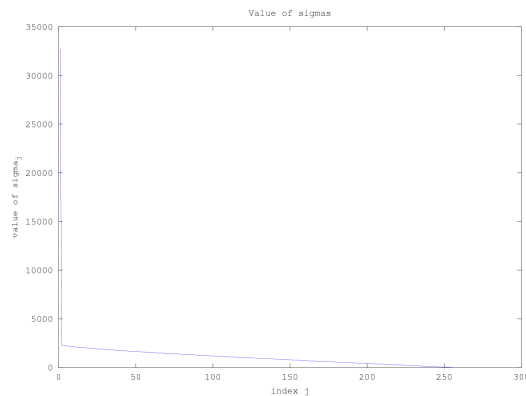
Vi kan se at verdien for σ_j raskt går mot 0. Dette betyr at den viktige informasjonen om bildet ligger i relativt få σ 'er. Som vi så i deloppgave c fikk vi et ganske klart bilde med bare de 32 første σ 'ene.

For en tilfeldig matrise får vi dette bildet:



Figur 4: Bildet som tilsvarer en tilfeldig matrise. Bare støy.

Vi får da denne fordelingen av σ



Figur 5: Sigmaverdiene for en tilfeldig matrise.

Vi kan se at σ_1 inneholder det meste av informasjonen. Men etter dette synker verdiene for σ saktere enn for Marilyn Monroe, hvilket betyr at vi trenger flere ledd for å gi en god tilnærming til den tilfeldige matrisen. Grunnet til dette kan være at bildet av Marilyn Monroe inneholder områder bestående av mye hvit rundt henne, i tillegg til at det er flere områder i ansiktet hennes med lik farge. I det tilfeldige bildet, derimot, er det ingen større områder med samme farge, og man trenger derfor flere σ 'er for å tilnærme det.

Oppgave 5

a) Om vi vil lagre bildet for k σ 'er må vi lagre disse σ 'er. I tillegg trenger vi å lagre k \mathbf{u}_j , som har størrelsen $m \times 1$, og k \mathbf{v}_j , som har størrelsen $1 \times n$. Vi trenger med andre ord $(k + k \cdot m + k \cdot n) \cdot 8$ bits.

Etter litt eksprementering fant jeg ut at $k = 40$ gir et ganske godt bilde. Det ser fortsatt litt kornete ut, men forstyrrelsene rundt nakken og håret har forsvunnet:



Figur 6: Et ganske bra bildet med $k = 40$

b) Programmet for å regne ut feilen:

```
function e=relError(A,AK)
A = double(A);
AK = double(AK);
e = norm(A-AK)/norm(A);
```

Feilen mellom originalbildet og tilnærmingen med $k = 40$ er:

```
>> relError(A,AK)
ans = 0.0064210
```