

POLITECHNIKA WROCŁAWSKA

INTELIGENCJA OBLICZENIOWA I JEJ ZASTOSOWANIA

Ćwiczenie 2
Metody redukcji wymiarowości
– nieujemna faktoryzacja macierzy
i dekompozycje tensorów

Autorzy:

Paweł ANDZIUL 200648
Robert CHOJNACKI 200685
Marcin SŁOWIŃSKI 200638

Prowadzący:

dr hab. inż. Rafał ZDUNEK

9 czerwca 2017

Spis treści

1	Zadanie 1	2
1.1	Algorytm ALS	2
1.2	Algorytm MUE	2
1.3	Algorytm HALS	2
1.4	Realizacja	2
1.5	Wyniki	5
2	Zadanie 2	6
3	Zadanie 3	6
3.1	Opis metody	6
3.2	Algorytm	6
3.3	Realizacja	7
3.4	Wyniki	10
4	Podsumowanie	14

1 Zadanie 1

Wygenerować faktory $A = [a_{ij}] \in R_+^{I \times J}$ i $X = [x_{jt}] \in R_+^{J \times T}$, gdzie $a_{ij} = \max(0, \check{a}_{ij})$ i $x_{jt} = \max(0, \check{x}_{jt})$ oraz $\check{a}_{ij}, \check{x}_{jt} \sim N(0, 1)$ (rozkład normalny). Wygeneruj syntetyczne obserwacje $Y = AX$ dla $I = 100, T = 1000, J = 10$. Stosując wybrane algorytmy NMF (ALS, MUE, HALS) wyznacz estymowane faktory \hat{A} i \hat{X} oraz unormowany błąd residualny w funkcji iteracji naprzemiennych. Oceń jakość estymacji stosując miary MSE (ang. Mean-Squared Error) lub SIR (ang. Signal-to-Interference Ratio).

1.1 Algorytm ALS

Algorytm ALS polega na N-krotnym powtórzeniu pętli, której zadaniem jest obliczenie tensorów A oraz X . Wynik obliczenia jednego tensora wpływa na wynik drugiego, co dla kolejnych iteracji zmniejsza różnice między obrazem właściwym a zredukowanym.

Algorytm wykorzystuje następujące wzory:

$$\nabla_A D(Y|AX) = (AX - Y)X^T = 0 \Rightarrow AX X^T = Y X^T \Rightarrow A = Y X^T (X X^T)^{-1}$$

$$\nabla_X D(Y|AX) = A^T (AX - Y) = 0 \Rightarrow A^T AX = A^T Y \Rightarrow X = (A^T A)^{-1} A^T Y$$

Otrzymane tensory po każdej z iteracji przemnażane są przez siebie otrzymując obraz zredukowany, który następnie jest przyrównywany do oryginału oznaczonego jako Y .

1.2 Algorytm MUE

Algorytm MUE działa podobnie do algorytmu ALS. Zmieniają się w nim wzory odpowiadające za poszczególne tensory.

$$a_{ij} = a_{ij} \frac{[Y X^T]_{ij}}{[A X X^T]_{ij}}$$

$$x_{jt} = x_{jt} \frac{[A^T Y]_{jt}}{[A^T A]_{jt}}$$

Zaletą algorytmu MUE jest niski koszt obliczeniowy. Niestety, jest on obciążony kosztem w postaci powolnej zbieżności, co oznacza konieczność wykonania większej ilości obliczeń, które mogą zanegować jego zalety.

1.3 Algorytm HALS

Algorytm HALS od algorytmu ALS różni się iterowaniem osobno po wierszach i kolumnach.

$$a_j = [a_j + \frac{[Y X^T]_{*j} - A [X X^T]_{*j}}{[X X^T]_{jj}}]_+$$

$$x_j = [x_j + \frac{[A^T Y]_{j*} - [A^T A]_{j*} X}{[A^T A]_{jj}}]_+$$

Algorytm ten charakteryzuje się najmniejszym błędem residualnym osiąganym już przy niskiej ilości iteracji.

1.4 Realizacja

Listing 1: Skrypt wywołujący realizacje w środowisku MATLAB

```
1 clc;
2 clear all;
3 close all;
4
5 % dane oryginalne
```

```

6 I = 100; T = 1000; J = 10;
7 Aw = max(0,randn(I,J));
8 Xw = max(0,randn(J,T));
9
10 Y = Aw*Xw;
11
12 % inicjalizacja
13 A = rand(size(Y,1),J);
14 X = rand(J,size(Y,2));
15
16 A1 = A;
17 A2 = A;
18 A3 = A;
19
20 X1 = X;
21 X2 = X;
22 X3 = X;
23
24 MaxIter = 100;
25
26 [A1,X1,res1] = NMF_ALS(A1,X1,Y,MaxIter);
27 [A2,X2,res2] = NMF_MUE(A2,X2,Y,MaxIter);
28 %[A3,X3,res3] = NMF_HALS(A3,X3,Y,J,MaxIter);
29
30 figure
31 hold on;
32 semilogy(res1)
33 semilogy(res2)
34 %semilogy(res3)
35 legend('ALS', 'MUE')
36
37 % to samo ale z wykorzystaniem wbudowanej funkcji
38
39 rng(1) % for reproducibility
40 [W,H] = nnmf(Y,10);
41 D = norm(Y-W*H,'fro')/sqrt(I*T); % blad residualny (resztowy)
42
43
44 SIR = CalcSIR(Aw,A);
45
46 % ---- to co pisal na konsoli----
47
48 Aws = Aw*diag(1./sum(Aw,1));
49 grid on
50 eps
51 Aws(1:5,:)

```

Listing 2: Algorytm ALS

```

1 function[A,X,res] = NMF_ALS(A,X,Y,N)
2
3     for k = 1:N
4         % obliczanie A
5         A = max(0,Y*X'*inv(X*X'));
6         A = A*diag(1./sum(A,1));

```

```

7
8      % obliczanie X
9      X = max(0,inv(A'*A)*A'*Y);
10
11     % blad residualny
12     res(k) = norm(Y - A*X,'fro')/norm(Y,'fro');
13
14 end
15 end

```

Listing 3: Algorytm MUE

```

1 function [A,X,res] = MUE(A,X,Y,N)
2 for k = 1:N
3     % obliczanie A
4     s1 = A.*(Y*X');
5     s2 = (A*X*X');
6
7     A = max(0, s1./s2);
8     A = A*diag(1./sum(A,1));
9
10    % obliczanie X
11    s1 = X.*(A'*Y);
12    s2 = A'*A*X;
13
14    X = max(0, s1./s2);
15
16    % blad residualny
17    res(k) = norm(Y - A*X, 'fro')/norm(Y, 'fro');
18 end

```

Listing 4: Algorytm HALS

```

1 function [A,X,res] = NMF_HALS(A,X,Y,J,MaxIter)
2 for k = 1:MaxIter
3     for j = 1:J
4         % obliczanie A - po kolumnach
5         aj = A(:,j);
6
7         s1 = (Y*X');
8         s1 = s1(:,j);
9
10        s2 = A*(X*X');
11        s2 = s2(:,j);
12
13        s3 = (X*X');
14        s3 = s3(j:j);
15
16        combined = (aj + ((s1 - s2) / s3));
17
18        A(:,j) = max(0, combined);
19        A = A*diag(1./sum(A,1));
20    end
21    for j = 1:J
22        % obliczanie X - po wierszach

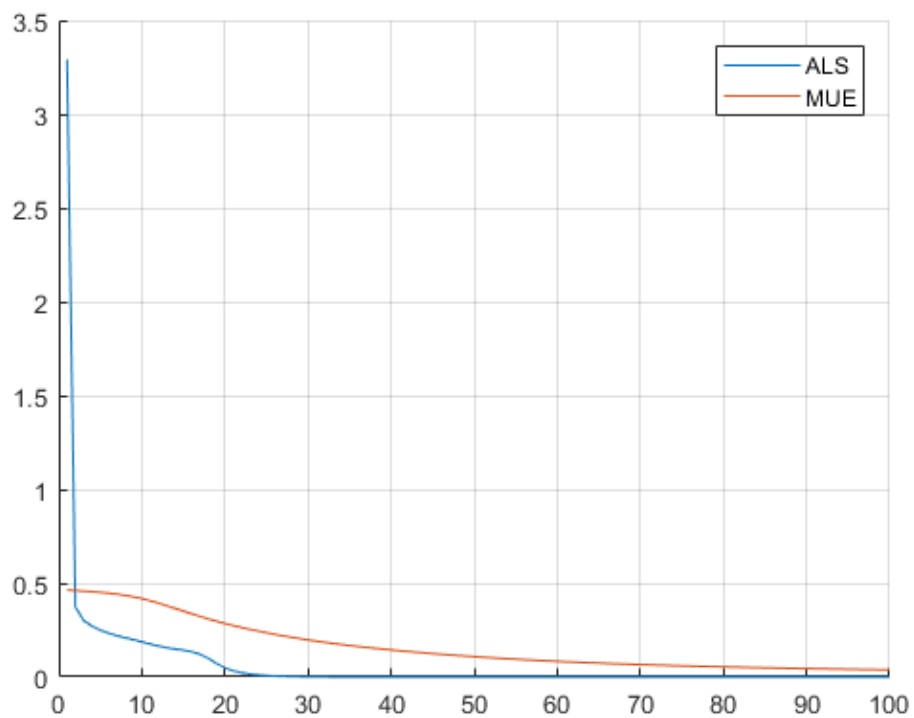
```

```

23     xj = X(j,:);
24
25     s1 = (A'*Y);
26     s1 = s1(j,:);
27
28     s2 = (A'*A);
29     s2 = s2(j,:);
30     s2 = s2*X;
31
32     s3 = (A'*A);
33     s3 = s3(j:j);
34
35     combined(xj + ((s1-s2)/s3));
36
37     X(j,:) = max(0, combined);
38 end
39 % blad residualny
40 res(k) = norm(Y - A*X, 'fro')/norm(Y,'fro');
41 end
42 end

```

1.5 Wyniki



Rysunek 1: Wykres ilustrujący przebieg optymalizacji naprzemiennej

2 Zadanie 2

Wygenerować faktory..

3 Zadanie 3

Obrazy twarzy z bazy ORL (lub podobnej) przedstaw za pomocą tensora $Y \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, gdzie I_3 jest liczbą obrazów. Rozdziel obrazy na zbiory trenujący i testujący według odpowiedniej zasady, np. 5-folds CV i utwórz odpowiednie tensory trenujący Y_r i testujący Y_t . Tensor trenujący poddaj dekompozycji CP (np. algorytmem ALS) oraz HOSVD dla $J = 4, 10, 20, 30$. Pogrupować obrazy stosując metodę k-średnich dla faktora $\hat{U}^{(3)}$. Badania przeprowadzić dla różnej liczby grup. Porównać dokładność grupowania z metodą PCA (z poprzedniego ćwiczenia). Następnie dokonaj projekcji obrazów z tensora Y_t na podprzestrzeń cech generowaną faktorami otrzymanymi z Y_r . Dokonaj klasyfikacji obrazów w przestrzeni cech w $\hat{U}^{(3)}$ za pomocą klasyfikatora k-NN. Porównać efekty klasyfikacji różnymi metodami (np. PCA, CP, HOSVD).

3.1 Opis metody

3.2 Algorytm

3.3 Realizacja



Rysunek 2: Twarze wykorzystane podczas testów

Listing 5: Podstawowy skrypt z realizacją w środowisku MATLAB

```
1 clc;
2 clear;
3 close all;
4 load('FaceData_56_46.mat');
5
6 Persons = 8;
7 ImagesPerPerson = 10;
8 nOfImages = Persons*ImagesPerPerson;
9
10 % wczytanie danych do tensora
11 P = zeros(1,nOfImages);
12 Y=zeros(56,46,nOfImages);
13 img_index = 1;
14 for p=(1:Persons)
15     for i=(1:ImagesPerPerson)
16         P(img_index) = p;
```



```

17         Y(:,:,img_index) = FaceData(p, i).Image;
18         img_index = img_index + 1;
19     end
20 end
21 P = P';
22
23 figure;
24 subtitle('Twarze oryginalne');
25 for i=(1:nOfImages)
26     subplot(Persons, ImagesPerPerson, i);
27     imagesc(Y(:,:,i));
28     title(i)
29     colormap gray;
30     set(gca, 'XtickLabel', [], 'YtickLabel', []);
31 end
32
33 % rozdzielenie na dwa zbiory (5-folds CV)
34 CV = cvpartition(P, 'kfold', 5);
35 train_idx = CV.training(1);
36 test_idx = CV.test(1);
37
38 % utworzenie tensorow trenujacego i testowego
39 Y_train = Y(:,:,train_idx);
40 Y_test = Y(:,:,test_idx);
41 Class_train_idx = P(train_idx);
42 Class_test_idx = P(test_idx);
43
44 J_serie = [4 10 20 30];
45
46 res_rands = zeros(1,length(J_serie));
47 res_time_all = zeros(1,length(J_serie));
48 res_time_train = zeros(1,length(J_serie));
49 res_acc = zeros(1,length(J_serie));
50 res_delta = zeros(1,length(J_serie));
51 res_groups_kmeans = [];
52
53 for J_current=(1:length(J_serie))
54     J(1:3) = J_serie(J_current);
55
56     % dekompozycja hosvd (pod kmeansa)
57     tic
58     [A, B, C, G, Y_hat] = skrypt_zad3_hosvd(Y, J);
59     res_time_all(J_current) = toc;
60
61     figure;
62     subtitle(sprintf('Twarze zredukowane J=%d (HOSVD)', J_serie(J_current)));
63     for i=(1:nOfImages)
64         subplot(Persons, ImagesPerPerson, i);
65         imagesc(Y_hat(:,:,i));
66         title(i)
67         colormap gray;
68         set(gca, 'XtickLabel', [], 'YtickLabel', []);
69     end
70
71     % grupowanie metoda ksrednich dla faktora U^(3) - stala liczba grup (ilosc
       osob)

```

```

72 kmeans_result = kmeans(C, Persons);
73 res_groups_kmeans = [res_groups_kmeans kmeans_result];
74 [res_acc(J_current), res_rands(J_current), ~] = AccMeasure(P,
    kmeans_result');
75
76 % dekompozycja hosvd
77 tic
78 [Ar, Br, Cr, Gr, Yr_hat] = skrypt_zad3_hosvd(Y_train, J);
79 res_time_train(J_current) = toc;
80
81 % projekcja
82 Y3 = reshape(permute(Y_test,[3 1
    2]),size(Y_test,3),size(Y_test,1)*size(Y_test,2));
83 G3 = reshape(permute(Gr,[3 1 2]),[J(3),J(1)*J(2)]);
84 Ct = Y3*pinv(double(G3)*(kron(Br,Ar))');
85 Ct = Ct.*repmat(1./sqrt(sum(Ct.^2,2)+eps),1,size(Ct,2));
86
87 % klasyfikacja w przestrzeni cech  $U^{(3)}$ 
88 mdl_class = fitcknn(Cr,Class_train_idx,'NumNeighbors',1);
89 prediction = predict(mdl_class, Ct);
90
91 % dokladnosc klasyfikacji
92 res_delta(J_current) = 100*(length(find((prediction -
    Class_test_idx)==0))/length(Class_test_idx));
93
94 end

```

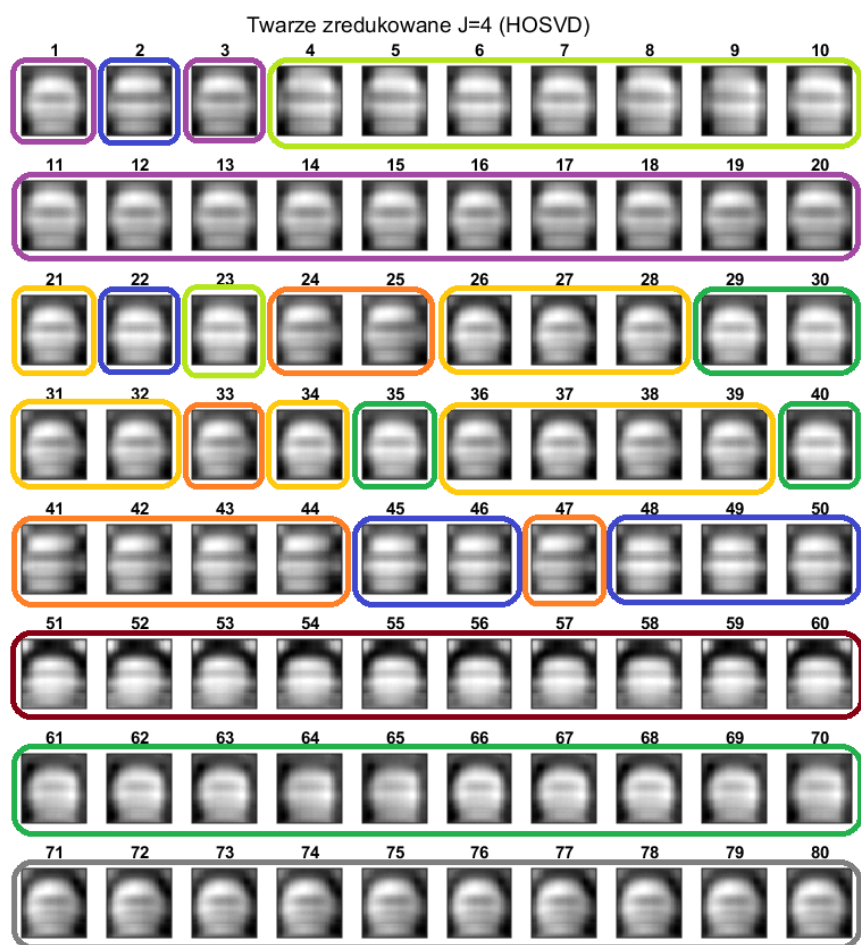
Listing 6: Funkcja realizująca algorytm HOSVD

```

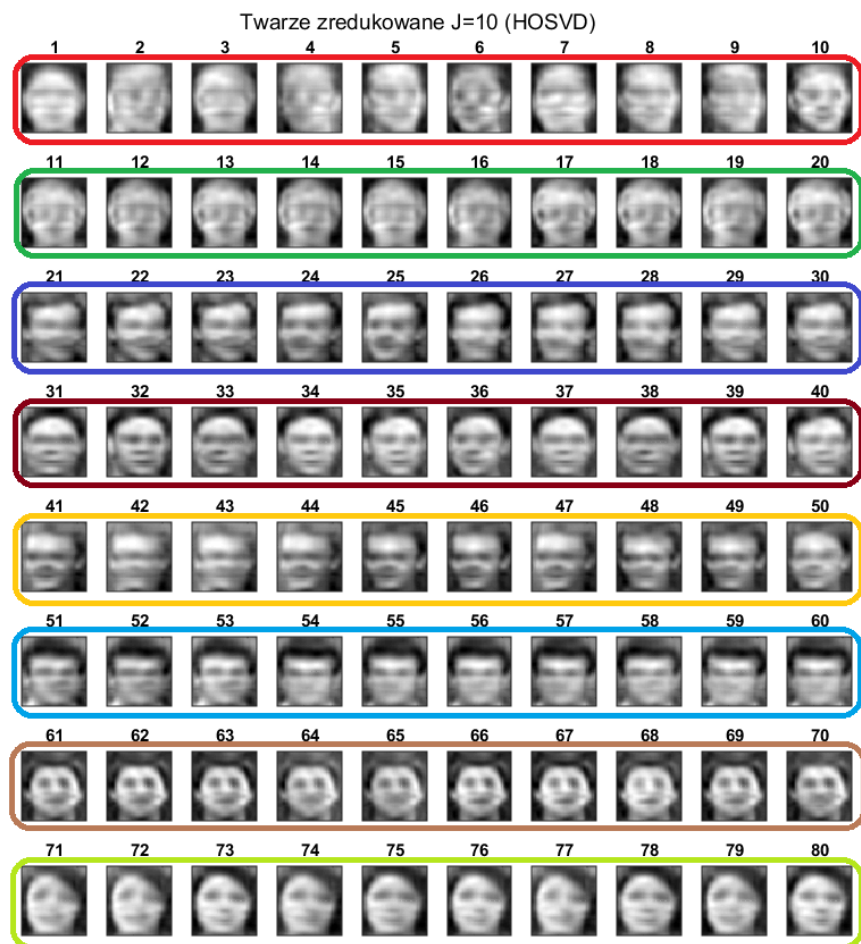
1 function [ A, B, C, G, Y_hat ] = skrypt_zad3_hosvd( Y, J )
2
3 DimY = size(Y);
4
5 % unfolding
6 Y1 = reshape(Y,DimY(1),DimY(2)*DimY(3));
7 Y2 = reshape(permute(Y,[2 1 3]),DimY(2),DimY(1)*DimY(3));
8 Y3 = reshape(permute(Y,[3 1 2]),DimY(3),DimY(1)*DimY(2));
9
10 % dekompozycja tensorow
11 [E1,~] = eig(Y1*Y1');
12 A = fliplr(E1(:,DimY(1)-J(1)+1:DimY(1)));
13
14 [E2,~] = eig(Y2*Y2');
15 B = fliplr(E2(:,DimY(2)-J(2)+1:DimY(2)));
16
17 [E3,~] = eig(Y3*Y3');
18 C = fliplr(E3(:,DimY(3)-J(3)+1:DimY(3)));
19
20 G = ntimes(ntimes(ntimes(Y,A',1,2),B',1,2),C',1,2); % core tensor
21 Y_hat = ntimes(ntimes(ntimes(G,A,1,2),B,1,2),C,1,2); % tensor 3-way
22
23 C = C.*repmat(1./sqrt(sum(C.^2,2)+eps),1,size(C,2));
24
25 end

```

3.4 Wyniki



Rysunek 3: Twarze zredukowane J=4 (HOSVD)



Rysunek 4: Twarze zredukowane J=10 (HOSVD)



Rysunek 5: Twarze zredukowane J=20 (HOSVD)



Rysunek 6: Twarze zredukowane J=4 (HOSVD)

W tabeli 1 zamieszczono otrzymane wartości metryk. Acc oraz Rand's index zostały policzone dla algorytmu k-średnich natomiast dokładność (delta) jest związana z klasyfikacją w przestrzeni cech $\hat{U}^{(3)}$ przy pomocy metody najbliższych sąsiadów.

Tabela 1: Otrzymane metryki dla różnych wartości parametru J

	4	10	20	30
Acc	76,25	100	80,00	63,75
Rand's index	92,56	100	93,04	84,56
delta	68,75	93,75	100	100

Jak możemy zauważyć metoda k-średnich daje najlepsze rezultaty dla J=10, zarówno mniej jak i więcej szczegółów w obrazie negatywnie wpływa na rezultat grupowania. W przypadku metody najbliższych sąsiadów jest inaczej – tutaj im bardziej szczegółowy obraz otrzyma ta metoda na wejściu tym lepsze będą rezultaty. Trzeba również pamiętać, że metoda k-średnich nie wymaga zbioru treningowego i uczącego.

4 Podsumowanie

..

Literatura

- [1] Dokumentacja środowiska MATLAB, <https://www.mathworks.com/>
- [2] Zdunek, Rafał, „Nieujemna faktoryzacja macierzy i tensorów : zastosowanie do klasyfikacji i przetwarzania sygnałów”, Oficyna Wydawnicza Politechniki Wrocławskiej, 2014
- [3] <http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html>
- [4] <http://www.esat.kuleuven.be/sista/tensorlab/>
- [5] <http://www.bsp.brain.riken.jp/TDALAB/>
- [6] <http://www.bsp.brain.riken.jp/~phan/>