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

12 czerwca 2017

Spis treści

1 Zadanie 1

Wygenerować faktory $A = [a_{ij}] \in R_{+}^{IxJ}$ i $X = [x_{jt}] \in R_{+}^{JxT}$, 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 Â i X̂ oraz unormowany błąd residualny w funkcji iteracji naprzemiennych. Oceń jakość estymacji stosując miary MSE (ang. Mean-Squarred Error) lub SIR (ang. Signal-to-Interference Ratio).

1.1 Metoda NMF

NMF (ang. Non-negative Matix Factorization) to grupa algorytmów z dziedziny wielowymiarowej analizy i algebry liniowej. Sama metoda zazwyczaj ogranicza się do liniowego modelu dekompozycji macierzy nieujemnej (faktoryzacji) na iloczyn dwóch macierzy o nieujemnych elementach, zwanych faktorami. Faktory mają różna interpretację w zależności od ich zastosowania, z których można wymienić: analizę skupień, nadzorowaną klasyfikację, ślępą separację źródeł itp.

Nieujemna faktoryzacja tensora jest multiliniowym rozszerzeniem metody NMF, pozwalającym na ekstrakcję rzadkich i nieujemnych wektorów, reprezentujących ukryte cechy tensora względem każdego z modów.

1.2 Algorytm ALS

Algorytm ALS (ang. *Alternating Least Squares*) 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 bazuje na wzorach 1 oraz 2.

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

$$\nabla_X D(Y|AX) = A^T(AX - Y) = 0 \Rightarrow A^TAX = A^TY \Rightarrow X = (A^TA)^{-1}A^TY \tag{2}$$

Otrzymane tensory po każdej z iteracji przemnażane są przez siebie otrzymując obraz zredukowany, który następnie jest przyrównywany do oryginału oznaczonego jako Y. Złożoność obliczeniową algorytmu można zgrubnie oszacować jako $O(J^3)$.

1.3 Algorytm MUE

Algorytm MUE (ang. *Multiplicative Updates for Euclidean distance*) działa podobnie do algorytmu ALS. Zmieniają się w nim wzory odpowiadające za poszczególne tensory.

$$a_{ij} = a_{ij}\frac{[YX^T]_{ij}}{[AXX^T]_{ij}} \tag{3}$$

$$x_{jt} = x_{jt}\frac{[A^TY]_{jt}}{[A^TAX]_{jt}} \tag{4}$$

Zaletą algorytmu MUE jest mała złożoność obliczeniowa, którą można oszacować jako $O(IJT + J^2I + J^2T)$. Niestety, jest on obarczony kosztem w postaci powolnej zbieżności, co oznacza konieczność wykonania większej ilości obliczeń, które mogą zanegować jego zalety.

1.4 Algorytm HALS

Algorytm HALS (ang. *Hierarchical ALS*) jest algorytmem blokowo-sekwencyjnym. Od standardowego algorytmu ALS różni się tym, że w danym kroku iteracyjnym aktualizowana jest tylko pojedyncza zmienna.

$$a_j = [a_j + \frac{[YX^T]_{*j} - A[XX^T]_{*j}}{[XX^T]_{jj}}]_+ \tag{5}$$

$$x_j = [x_j + \frac{[A^TY]_{j*} - [A^TA]_{j*}X}{[A^TA]_{jj}}]_+ \tag{6}$$

Algorytm ten charakteryzuje się najmniejszym błędem residualnym osiąganym już przy niskiej ilości iteracji. Złożoność obliczeniową można oszacować jako $O(kJ^2T) + O(J^2I + IJT)$.

1.5 Realizacja

Na poniższych listingach zamieszczono praktyczną implementację algorytmów w środowisku MATLAB.

Listing 1: Skrypt wywołujący realizacje w środowisku MATLAB

```
clc;
clear all;
close all;

% dane oryginalne
I = 100; T = 1000; J = 10;
Aw = max(0,randn(I,J));
Xw = max(0,randn(J,T));

Y = Aw*Xw;

% inicjalizacja
A = rand(size(Y,1),J);
X = rand(J,size(Y,2));

A1 = A; A2 = A; A3 = A;
X1 = X; X2 = X; X3 = X;

MaxIter = 300;
MSE_ALS = []; MSE_MUE = []; MSE_HALS = [];
[A1,X1,res1,MSE_ALS] = skrypt_zad1_nmf_als(A1,X1,Y,MaxIter);
[A2,X2,res2,MSE_MUE] = skrypt_zad1_nmf_mue(A2,X2,Y,MaxIter);
[A3,X3,res3,MSE_HALS] = skrypt_zad1_nmf_hals(A3,X3,Y,J,MaxIter);

figure
semilogy(res1);
hold on;
semilogy(res2);
semilogy(res3);
legend('ALS', 'MUE', 'HALS');
title('Blad residualny');
hold off;
grid on;
```

```

figure;
hold on
s1 = semilogy(MSE_ALS);
s2 = semilogy(MSE_MUE);
s3 = semilogy(MSE_HALS);
legend([s1, s2, s3], {'ALS', 'MUE', 'HALS'})
title('Blad sredniokwadratowy');
set(gca, 'YScale', 'log')
hold off;
grid on;

SIR_A1 = CalcSIR(A',A1');
SIR_A2 = CalcSIR(A',A2');
SIR_A3 = CalcSIR(A',A3');

figure;
hold on
title('Wartosci SIR');
plot(1:100, SIR_A1, 1:100, SIR_A2, 1:100, SIR_A3);
xlim([1 100]);
legend('ALS','MUE','HALS', 'Location','southeast','Orientation','vertical')
ylabel('SIR')
hold off
```

Listing 2: Algorytm ALS

```
function[A,X,res,MSE] = skrypt_zad1_nmf_als(A,X,Y,N)
    MSE = [];
    for k = 1:N
        % obliczanie A
        A = max(0,Y*X'*inv(X*X'));
        A = A*diag(1./sum(A,1));

        % obliczanie X
        X = max(0,inv(A'*A)*A'*Y);

        % blad residualny
        res(k) = norm(Y - A*X,'fro')/norm(Y,'fro');

        % blad srednio-kwadratowy
        MSE = [MSE immse(Y,A*X)];
    end
end
```

Listing 3: Algorytm MUE

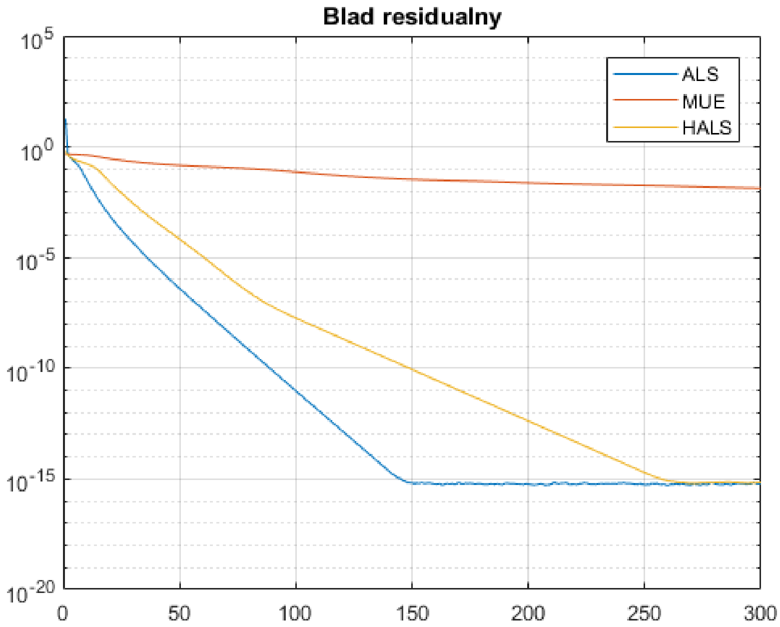
```
function [A,X,res,MSE] = skrypt_zad1_nmf_mue(A,X,Y,N)
    MSE = [];
    for k = 1:N
        % obliczanie A
        s1 = A.*(Y*X');
        s2 = (A*X*X');
        A = max(0, s1./s2);
        A = A*diag(1./sum(A,1));

        % obliczanie X
        s1 = X.*(A'*Y);
        s2 =A'*A*X;
        X = max(0, s1./s2);

        % blad residualny
        res(k) = norm(Y - A*X, 'fro')/norm(Y,'fro');

        % blad srednio-kwadratowy
        MSE = [MSE immse(Y,A*X)];
    end
end
```

Listing 4: Algorytm HALS

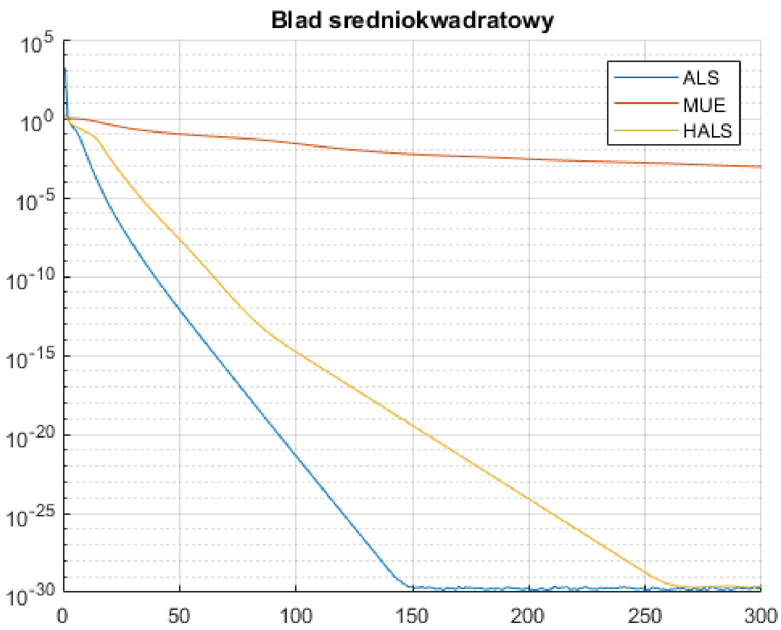
```
function [A,X,res,MSE] = skrypt_zad1_nmf_hals(A,X,Y,J,MaxIter)
    MSE = [];
    for k = 1:MaxIter
        for j = 1:J % obliczanie A - po kolumnach
            YXp = Y*X';
            XXp = X*X';
            A(:,j) = max(0, A(:,j)+(YXp(:,j)-A*XXp(:,j))/(XXp(j,j)+eps));
            A = A*diag(1./sum(A,1));
        end
        for j = 1:J % obliczanie X - po wierszach
            ApY = A'*Y;
            ApA = A'*A;
            X(j,:) = max(0, X(j,:)+(ApY(j,:)-ApA(j,:)*X)/(ApA(j,j)+eps));
        end
        res(k) = norm(Y - A*X, 'fro')/norm(Y,'fro'); % blad residualny
        % blad srednio-kwadratowy
        MSE = [MSE immse(Y,A*X)];
    end
end
```

1.6 Wyniki

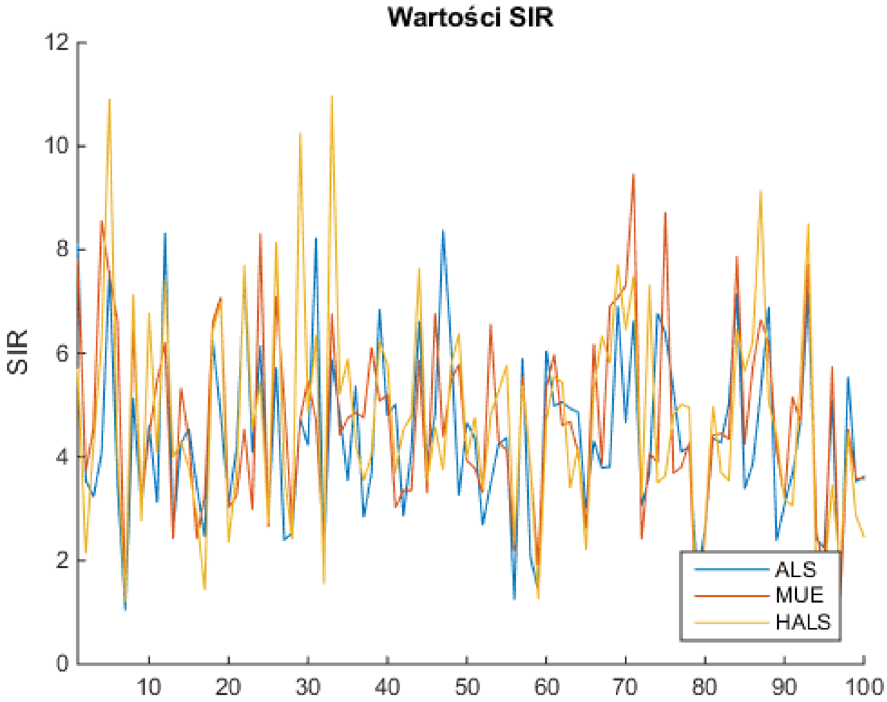
Na ilustracji 1 i 2 zamieszczono graficzne porównanie przebiegu optymalizacji kolejno algorytmami ALS, MUE oraz HALS.



Rysunek 1: Błąd residualny



Rysunek 2: Błąd średniokwadratowy



Rysunek 3: Wykres SIR

2 Zadanie 2

Wygenerować faktory $U^(1) = [u^{(1)}_{i_1j}] \in R^{I_1xJ}_+$, $U^(2) = [u^{(2)}_{i_2j}] \in R^{I_2xJ}_+$, $U^(3) = [u^{(3)}_{i_3j}] \in R^{I_3xJ}_+$, gdzie $[u^{(1)}_{i_1j}] = max(0, \breve{u}^{(1)}_{i_1j})$, $[u^{(2)}_{i_2j}] = max(0, \breve{u}^{(2)}_{i_2j})$, $[u^{(3)}_{i_3j}] = max(0, \breve{u}^{(3)}_{i_3j})$ oraz $\breve{u}^{(1)}_{i_1j}$, $\breve{u}^{(2)}_{i_2j}$, $\breve{u}^{(3)}_{i_3j}$ $\sim N(0, 1)$ (rozkład normalny). Wygeneruj syntetyczne obserwacje Y dla $I_1 = 10$, $I_2 = 20$, $I_3 = 30$, $J = 5$. Stosując wybrane algorytmy NTF (np. ALS) wyznacz estymowane faktory $\breve{U}^{(1)}, \breve{U}^{(2)}, \breve{U}^{(3)}$ 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).

3 Zadanie 3

Obrazy twarzy z bazy ORL (lub podobnej) przedstaw za pomocą tensora $Y \in R^{I_1xI_2xI_3}$, 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 $\widehat{U}^{(3)}$. 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 $\widehat{U}^{(3)}$ za pomocą klasyfikatora k-NN. Porównać efekty klasyfikacji różnymi metodami (np. PCA, CP, HOSVD).

3.1 Opis metody

HOSVD jest szczególnym przypadkiem modelu dekompozycji Tuckera. Kolumny każdej z macierzy czynnikowych (faktorów) są wzajemnie ortogonalne, a tensor rdzeniowy $\mathcal{G}$ nie jest superdiagonalny. Dokonujemy kolejno matrycyzacji względem n-tego modu.

3.2 Algorytm

Wejściem algorytmu HO-SVD są: tensor danych $Y \in R^{I_1 x I_2 x I_3}$ oraz rzędy faktoryzacji $J = [J_1, ..., J_N]$, natomiast wyjściem: estymowane wektory $U^{(n)}$ oraz tensor rdzeniowy $\mathcal{G}$.

Pseudokod 1 Algorytm HO-SVD

for $n = 1, ..., N$ **do**

  $Y_{(n)} \leftarrow$ unfolding($\mathcal{Y}$, n) $\in R^{I_n \times \prod_{p \neq n} I_p}$

  $U_{(n)} \leftarrow$ eigs$(Y_{(n)}(Y_{(n)})^T, J_n)$

Wyznacz tensor $\mathcal{G}$

3.3 Realizacja

Na kolejnych listingach zamieszczono implementację w środowisku MATLAB. Listing 5 stanowi podstawowy potok przetwarzania od wczytania danych po wyświetlenie wyników pomiarów. Na listingu 6 zamieszczono implementację metody HOSVD. W celu przeprowadzenia klasyfikacji kNN dokonano podziału zbioru wejściowego przy pomocy *cvpartition* 5-folds (podzielono 80 zdjęć na dwa zbiory składające się odpowiednio z 64 i 16 obrazów).

Listing 5: Podstawowy skrypt z realizacją w środowisku MATLAB

```
clc;
clear;
close all;
load('FaceData_56_46.mat');

Persons = 8;
ImagesPerPerson = 10;
nOfImages = Persons*ImagesPerPerson;

% wczytanie danych do tensora
P = zeros(1,nOfImages);
Y=zeros(56,46,nOfImages);
img_index = 1;
for p=(1:Persons)
    for i=(1:ImagesPerPerson)
        P(img_index) = p;
        Y(:,:,img_index) = FaceData(p, i).Image;
        img_index = img_index + 1;
    end
end
P = P';

figure;
suptitle('Twarze oryginalne');
for i=(1:nOfImages)
    subplot(Persons, ImagesPerPerson, i);
```

```
    imagesc(Y(:,:,i));
    title(i)
    colormap gray;
    set(gca,'XtickLabel',[],'YtickLabel',[]);
end

% rozdzielenie na dwa zbiory (5-folds CV)
CV = cvpartition(P,'kfold',5);
train_idx = CV.training(1);
test_idx = CV.test(1);

% utworzenie tensorow trenujacego i testowego
Y_train = Y(:,:,train_idx);
Y_test = Y(:,:,test_idx);
Class_train_idx = P(train_idx);
Class_test_idx = P(test_idx);

J_serie = [4 10 20 30];

res_time_hosvd = zeros(1,length(J_serie));
res_time_kmeans = zeros(1,length(J_serie));
res_time_knn = zeros(1,length(J_serie));
res_acc_kmeans = zeros(1,length(J_serie));
res_acc_knn = zeros(1,length(J_serie));
res_rands_kmeans = zeros(1,length(J_serie));
res_rands_knn = zeros(1,length(J_serie));
res_delta = zeros(1,length(J_serie));
res_groups_kmeans = [];

for J_current=(1:length(J_serie))
    J(1:3) = J_serie(J_current);

    % dekompozycja hosvd (pod kmeansa)
    tic
    [A, B, C, G, Y_hat] = skrypt_zad3_hosvd(Y, J);
    res_time_hosvd(J_current) = toc;

    figure;
    suptitle(sprintf('Twarze zredukowane J=%d (HOSVD)', J_serie(J_current)));
    for i=(1:nOfImages)
        subplot(Persons, ImagesPerPerson, i);
        imagesc(Y_hat(:,:,i));
        title(i)
        colormap gray;
        set(gca,'XtickLabel',[],'YtickLabel',[]);
    end

    % grupowanie metoda ksrednich dla faktora U^(3) - stala liczba grup (ilosc
        osob)
    tic
    kmeans_result = kmeans(C, Persons);
    res_time_kmeans(J_current) = toc;
    res_groups_kmeans = [res_groups_kmeans kmeans_result];
    [res_acc_kmeans(J_current), res_rands_kmeans(J_current), ~] = AccMeasure(P,
        kmeans_result');

```

```
    % dekompozycja hosvd
    [Ar, Br, Cr, Gr, Yr_hat] = skrypt_zad3_hosvd(Y_train, J);

    % projekcja
    Y3 = reshape(permute(Y_test,[3 1
        2]),size(Y_test,3),size(Y_test,1)*size(Y_test,2));
    G3 = reshape(permute(Gr,[3 1 2]),[J(3),J(1)*J(2)]);
    Ct = Y3*pinv(double(G3)*(kron(Br,Ar))');
    Ct = Ct.*repmat(1./sqrt(sum(Ct.^2,2)+eps),1,size(Ct,2));

    % klasyfikacja w przestrzeni cech U^(3)
    tic
    mdl_class = fitcknn(Cr,Class_train_idx,'NumNeighbors',1);
    prediction = predict(mdl_class, Ct);
    res_time_knn(J_current) = toc;
    [res_acc_knn(J_current), res_rands_knn(J_current), ~] =
        AccMeasure(prediction, Class_test_idx');

    % dokladnosc klasyfikacji (podobnie jak w AccMeasure)
    res_delta(J_current) = 100*(length(find((prediction -
        Class_test_idx)==0))/length(Class_test_idx));

end
%%
figure;
hold on
title('Czas przetwarzania dla roznych wartosci J');
plot(J_serie, res_time_hosvd, J_serie, res_time_kmeans, J_serie, res_time_knn);
xlim([4 30]);
legend('hosvd','kmeans','knnclassify',
    'Location','southeast','Orientation','vertical')
xlabel('J')
ylabel('czas przetwarzania [s]')
hold off
```

Listing 6: Funkcja realizująca algorytm HOSVD

```
function [ A, B, C, G, Y_hat ] = skrypt_zad3_hosvd( Y, J )
    DimY = size(Y);

    % unfolding
    Y1 = reshape(Y,DimY(1),DimY(2)*DimY(3));
    Y2 = reshape(permute(Y,[2 1 3]),DimY(2),DimY(1)*DimY(3));
    Y3 = reshape(permute(Y,[3 1 2]),DimY(3),DimY(1)*DimY(2));

    % dekompozycja tensorow
    [E1,~] = eig(Y1*Y1');
    A = fliplr(E1(:,DimY(1)-J(1)+1:DimY(1)));

    [E2,~] = eig(Y2*Y2');
    B = fliplr(E2(:,DimY(2)-J(2)+1:DimY(2)));

    [E3,~] = eig(Y3*Y3');
    C = fliplr(E3(:,DimY(3)-J(3)+1:DimY(3)));

    G = ntimes(ntimes(ntimes(Y,A',1,2),B',1,2),C',1,2); % core tensor
    Y_hat = ntimes(ntimes(ntimes(G,A,1,2),B,1,2),C,1,2); % tensor 3-way

    C = C.*repmat(1./sqrt(sum(C.^2,2)+eps),1,size(C,2));

end
```

3.4 Wyniki

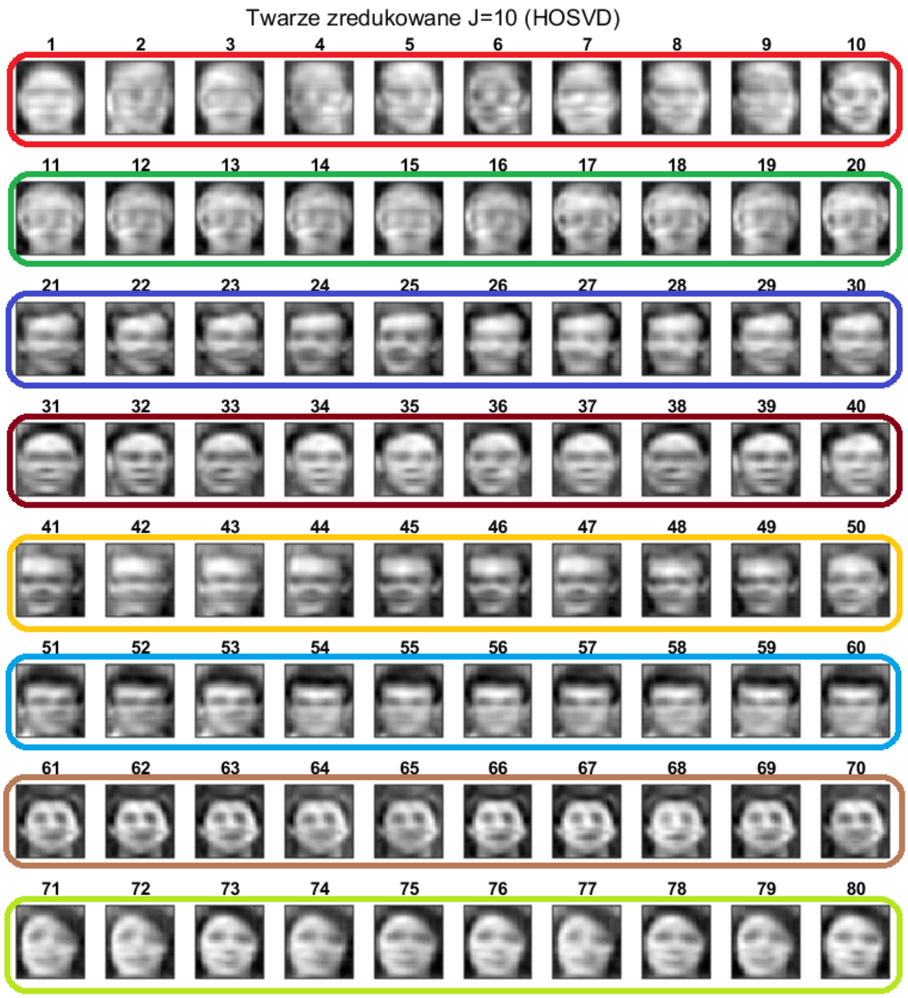
W niniejszym punkcie zamieszczono wyniki dla dekompozycji z wykorzystaniem metody HOSVD. Badaniu poddano 80 obrazów twarzy (ilustracja 4) z bazy Uniwersytetu Cambridge [7].

Rysunek 4: Twarze wykorzystane podczas testów

Ilustracja 5 przedstawia wynik grupowania metodą k-średnich dla J=4 oraz liczby grup równej liczbie osób wynoszącej 8. Cztery osoby zostały w całości zgrupowane poprawnie, najwięcej problemów sprawiły osoby 3 i 4, które okazały się zbyt podobne do siebie.

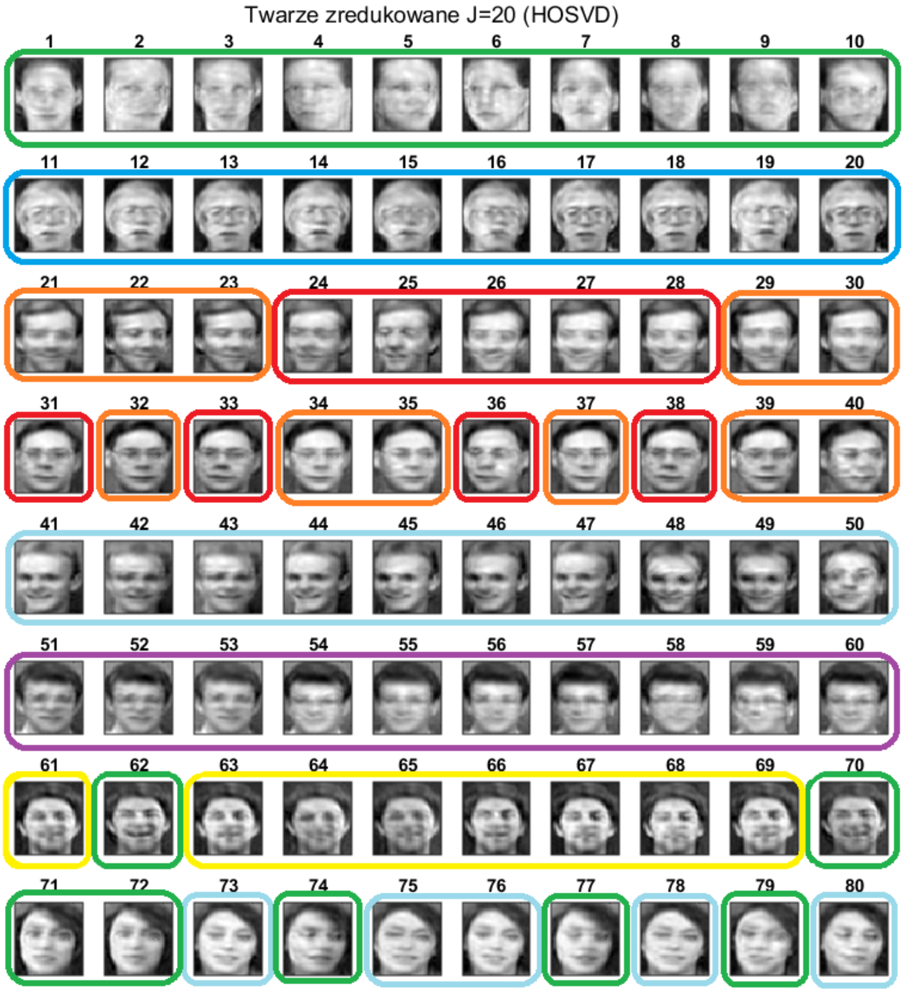
Rysunek 5: Twarze zredukowane J=4 (HOSVD)

Rysunek 6 przedstawia wynik grupowania metodą k-średnich dla J=10 oraz liczby grup równej liczbie osób wynoszącej 8. Metoda k-średnich poradziła tu sobie znakomicie. Wszystkie osoby zostały poprawnie zgrupowane.



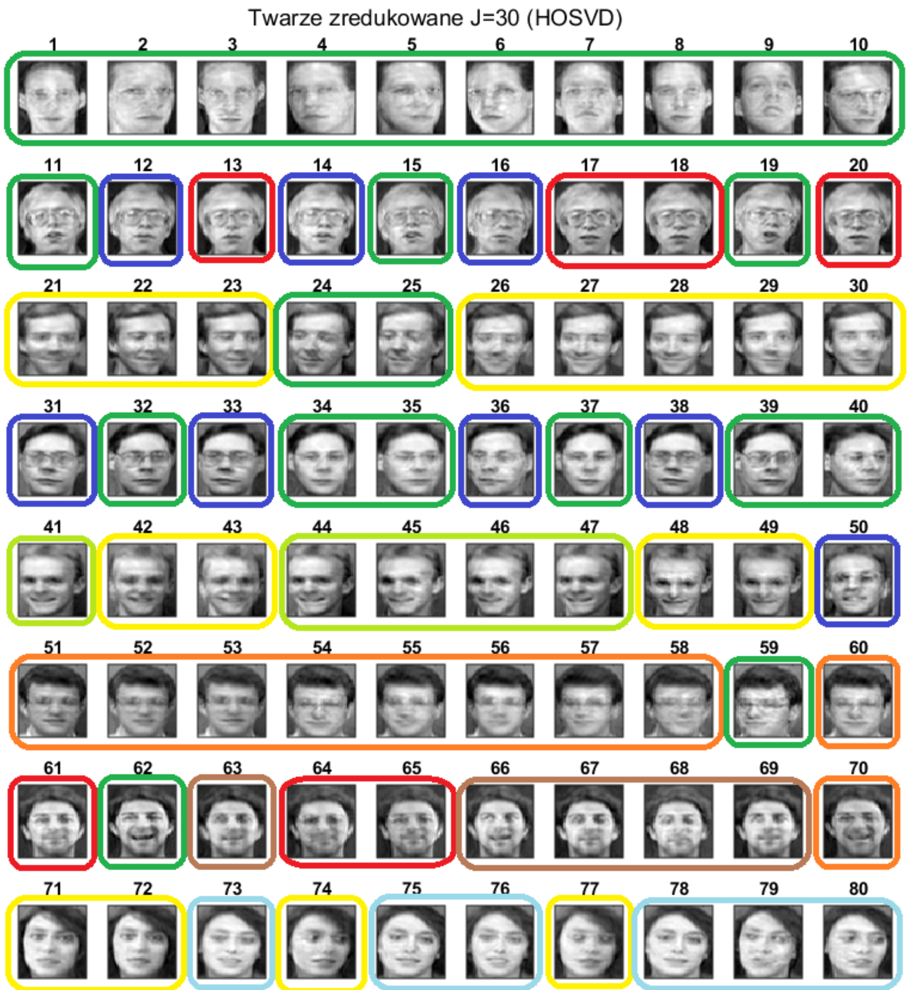
Rysunek 6: Twarze zredukowane J=10 (HOSVD)

Na ilustracji 7 przedstawiono wynik grupowania metodą k-średnich dla J=20 oraz liczby grup równej liczbie osób wynoszącej 8. Cztery osoby zostały zgrupowane poprawnie, osoby 3 i 4 zostały wzajemnie ze sobą pomieszane.



Rysunek 7: Twarze zredukowane J=20 (HOSVD)

Na rysunku 8 przedstawiono wynik grupowania metodą k-średnich dla J=30 oraz liczby grup równej liczbie osób wynoszącej 8. Wyniki dla tej próby są najgorsze, udało się zgrupować jedynie jedną osobę.



Rysunek 8: Twarze zredukowane J=30 (HOSVD)

W tabeli 1 zamieszczono otrzymane wartości metryk. Dla algorytmu k-średnich zostały policzone Acc (dokładność) oraz Rand's index.

Tabela 1: Otrzymane metryki dla różnych wartości parametru J (k-średnich)

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

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.

W tabeli 2 zamieszczono metryki dla klasyfikacji metodą najbliższych sąsiadów. Zostały policzone dokładność (Acc) oraz Rand's index dla klasyfikacji w przestrzeni cech $\widehat{U}^{(3)}$ przy pomocy metody najbliższych sąsiadów.

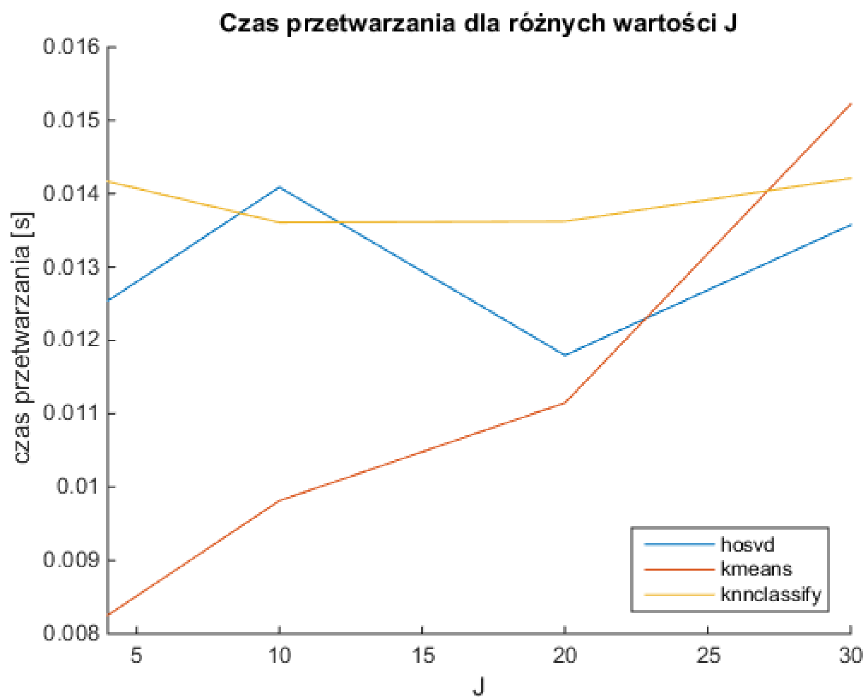
Tabela 2: Otrzymane metryki dla różnych wartości parametru J (k-najbliższych sąsiadów)

	4	10	20	30
Acc (dokładność)	81,25	93,75	100	100
Rand's index	94,17	97,50	100	100

Natomiast zależności czasowe przedstawiono w tabeli 3. Grupowanie przeprowadzono dla ilości grup równej liczbie osób aby móc jednoznacznie zinterpretować wyniki. Na ilustracji 9 zamieszczono graficzne porównanie.

Tabela 3: Czas przetwarzania [ms] w zależności od parametru J

	4	10	20	30
HOSVD	12,54	14,09	11,80	13,58
grupowanie k-średnich	8,25	9,81	11,15	15,22
klasyfikacja k-NN	14,17	13,61	13,62	14,21



Rysunek 9: Czas przetwarzania w zależności od wartości J

4 Podsumowanie

Podczas zajęć laboratoryjnych mieliśmy okazję zapoznać się z metodami redukcji wymiarowości przy pomocy nieujemnej faktoryzacji macierzy oraz technikami dekompozycji tensorów. Celem pierwszego zadania było zapoznanie z algorytmami stosowanymi do estymacji faktorów. O ile ich implementacja w środowisku MATLAB wydaje się prosta, o tyle wyznaczenie wymaga skomplikowanych obliczeń matematycznych. Otrzymane wyniki pozwoliły ocenić skuteczność każdego z poszczególnych algorytmów.

W przypadku analizy wyników z zadania trzeciego, zauważono, że zarówno mniejsza jak i większa liczba szczegółów w obrazie, za którą odpowiadał parametr J, negatywnie wpływa na rezultat grupowania metodą k-średnich. W przypadku metody najbliższych sąsiadów jest zupełnie odwrotnie – im większy parametr J, czyli bardziej szczegółowy obraz, tym lepsze są rezultaty.

Redukcja wymiarowości pozwala w znacznym stopniu ograniczyć rozmiar przetwarzanych danych przy niewielkiej stracie informacji. Pozwala to na uzyskanie zbliżonych efektów, jednak przy krótszym czasie wykonywania obliczeń.

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/`

[7] `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`