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 3
1.5 Wyniki 5

2 Zadanie 3 **6**
2.1 Opis metody 6
2.2 Algorytm 6
2.3 Realizacja 6
2.4 Wyniki 9

3 Podsumowanie **14**

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 $\hat{A}$ i $\hat{X}$ 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 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 bazuje na wzorach 1 oraz 2.

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

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

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.

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

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

Zaletą algorytmu MUE jest niski koszt obliczeniowy. Niestety, jest on obarczony 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.

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

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

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

1.4 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 = 100;

[A1,X1,res1] = NMF_ALS(A1,X1,Y,MaxIter);
[A2,X2,res2] = NMF_MUE(A2,X2,Y,MaxIter);
%[A3,X3,res3] = NMF_HALS(A3,X3,Y,J,MaxIter);

figure
semilogy(res1);
hold on;
semilogy(res2);
%semilogy(res3);
legend('ALS', 'MUE');
hold off;
grid on;

rng(1) % for reproducibility
[W,H] = nnmf(Y,10);
D = norm(Y-W*H,'fro')/sqrt(I*T); % blad residualny (resztowy)

SIR = CalcSIR(Aw,A);
```

Listing 2: Algorytm ALS

```
function[A,X,res] = NMF_ALS(A,X,Y,N)

    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');

    end
end
```

Listing 3: Algorytm MUE

```
function [A,X,res] = MUE(A,X,Y,N)
  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');
  end
```

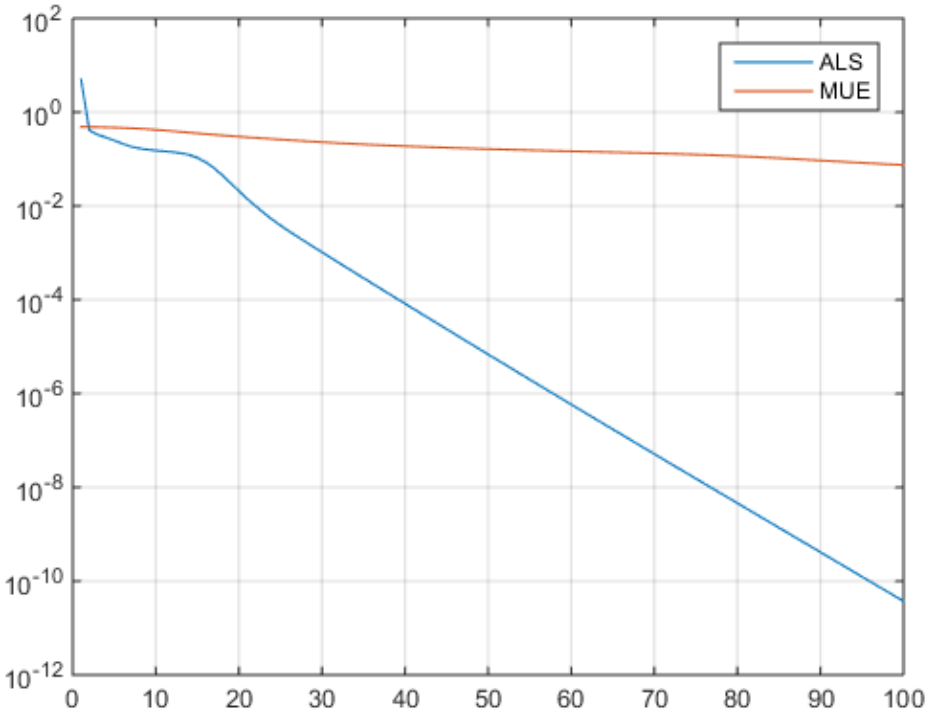
Listing 4: Algorytm HALS

```
function [A,X,res] = NMF_HALS(A,X,Y,J,MaxIter)
  for k = 1:MaxIter
    for j = 1:J
      % obliczanie A - po kolumnach
      aj = A(:,j);

      s1 = (Y*X');
      s1 = s1(:,j);

      s2 = A*(X*X');
      s2 = s2(:,j);

      s3 = (X*X');
      s3 = s3(j:j);

      combined = (aj + ((s1 - s2) / s3));

      A(:,j) = max(0, combined);
      A = A*diag(1./sum(A,1));
    end
    for j = 1:J
       % obliczanie X - po wierszach
       xj = X(j,:);

```

```
        s1 = (A'*Y);
        s1 = s1(j,:);

        s2 = (A'*A);
        s2 = s2(j,:);
        s2 = s2*X;

        s3 = (A'*A);
        s3 = s3(j:j);

        combined(xj + ((s1-s2)/s3));

        X(j,:) = max(0, combined);
    end
    % blad residualny
    res(k) = norm(Y - A*X, 'fro')/norm(Y,'fro');
  end
end
```

1.5 Wyniki

Na ilustracji 1 zamieszczono graficzne porównanie przebiegu optymalizacji kolejno algorytmami ALS oraz MUE.



Rysunek 1: Wykres ilustrujący przebieg optymalizacji ALS oraz MUE

2 Zadanie 3

Obrazy twarzy z bazy ORL (lub podobnej) przedstaw za pomocą tensora $Y \in R^{I_1 x I_2 x I_3}$, 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 $\widehat{U}^{(3)}$. 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).

2.1 Opis metody

2.2 Algorytm

2.3 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.

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');
```

```
80
81      % dekompozycja hosvd
82      [Ar, Br, Cr, Gr, Yr_hat] = skrypt_zad3_hosvd(Y_train, J);
83
84      % projekcja
85      Y3 = reshape(permute(Y_test,[3 1
            2]),size(Y_test,3),size(Y_test,1)*size(Y_test,2));
86      G3 = reshape(permute(Gr,[3 1 2]),[J(3),J(1)*J(2)]);
87      Ct = Y3*pinv(double(G3)*(kron(Br,Ar))');
88      Ct = Ct.*repmat(1./sqrt(sum(Ct.^2,2)+eps),1,size(Ct,2));
89
90      % klasyfikacja w przestrzeni cech U^(3)
91      tic
92      mdl_class = fitcknn(Cr,Class_train_idx,'NumNeighbors',1);
93      prediction = predict(mdl_class, Ct);
94      res_time_knn(J_current) = toc;
95      [res_acc_knn(J_current), res_rands_knn(J_current), ~] =
            AccMeasure(prediction, Class_test_idx');
96
97      % dokladnosc klasyfikacji (podobnie jak w AccMeasure)
98      res_delta(J_current) = 100*(length(find((prediction -
            Class_test_idx)==0))/length(Class_test_idx));
99
100 end
101 %%
102 figure;
103 hold on
104 title('Czas przetwarzania dla roznych wartosci J');
105 plot(J_serie, res_time_hosvd, J_serie, res_time_kmeans, J_serie, res_time_knn);
106 xlim([4 30]);
107 legend('hosvd','kmeans','knnclassify',
        'Location','southeast','Orientation','vertical')
108 xlabel('J')
109 ylabel('czas przetwarzania [s]')
110 hold off
```

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');
```

```
    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
```

2.4 Wyniki

W niniejszym punkcie zamieszczono wyniki dla dekompozycji z wykorzystaniem metody HOSVD. Badaniu poddano 80 obrazów twarzy (ilustracja 2) z bazy Uniwersytetu Cambridge [7].

Rysunek 2: Twarze wykorzystane podczas testów

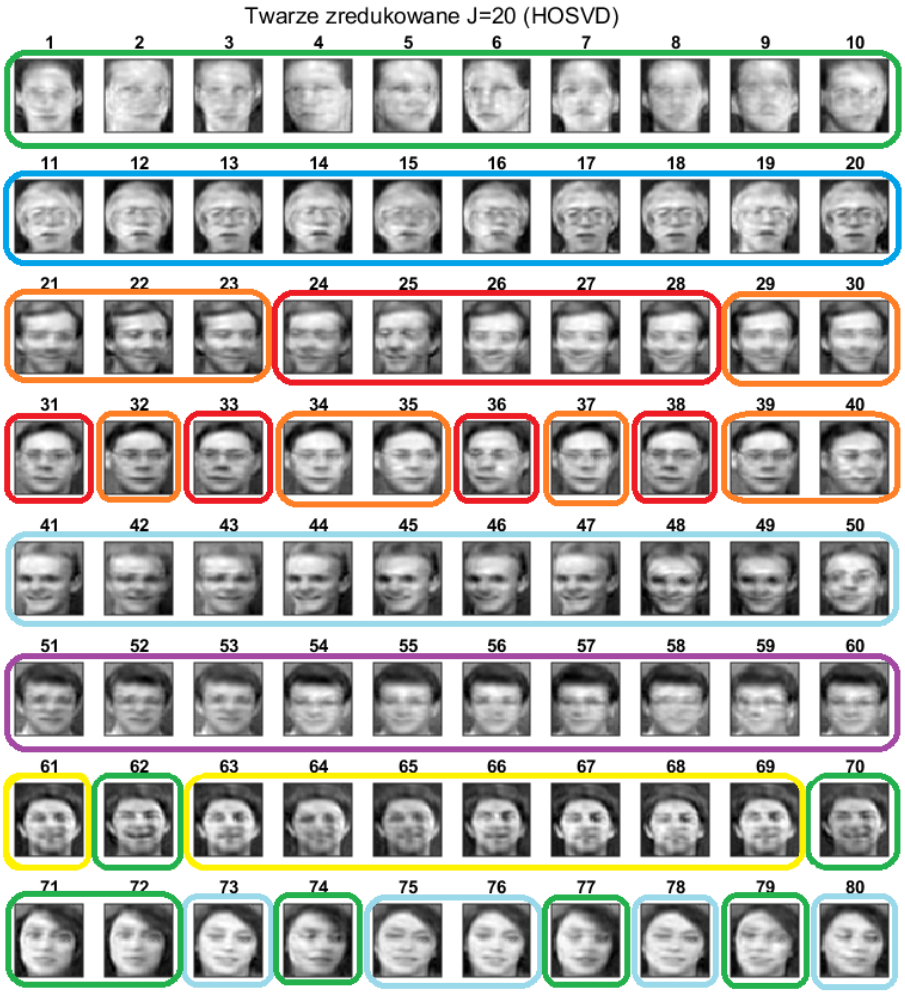
Ilustracja 3 przedstawia wynik grupowania metodą k-średnich dla J=4 oraz liczby grup równej liczbie osób wynoszącej 8.

Rysunek 3: Twarze zredukowane J=4 (HOSVD)

Rysunek 4 przedstawia wynik grupowania metodą k-średnich dla J=10 oraz liczby grup równej liczbie osób wynoszącej 8.

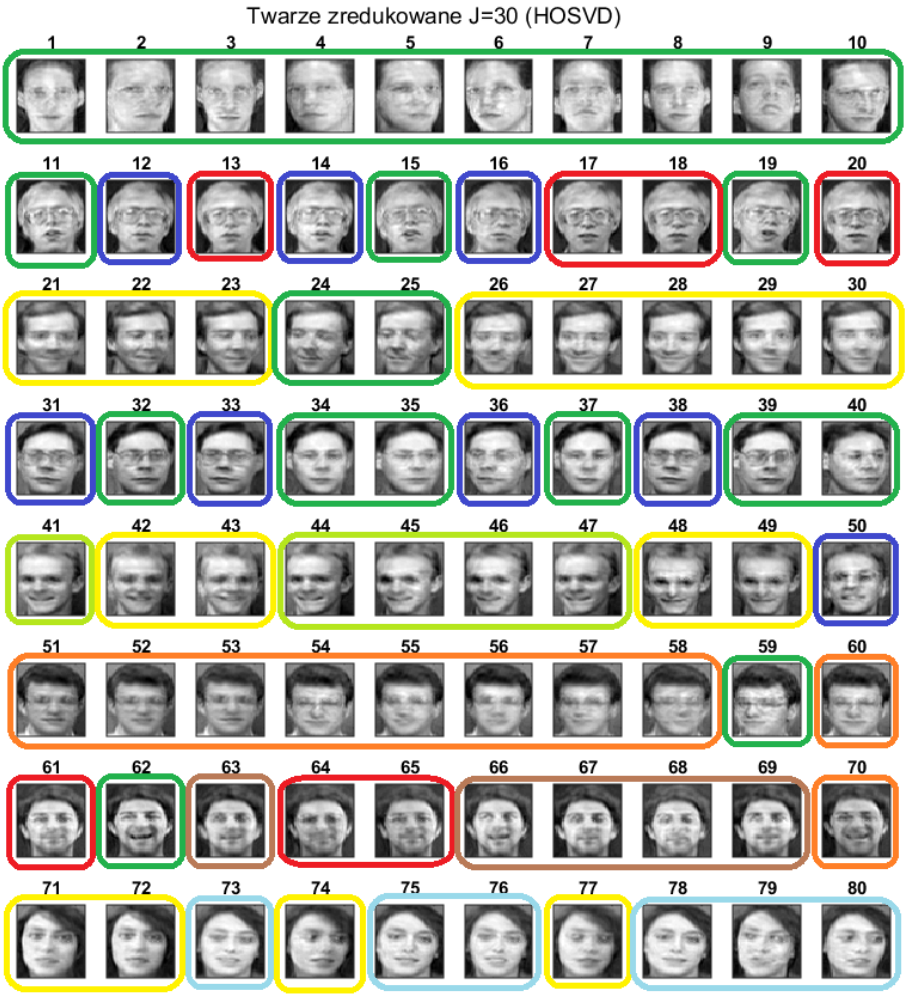
Rysunek 4: Twarze zredukowane J=10 (HOSVD)

Na ilustracji 5 przedstawiono wynik grupowania metodą k-średnich dla J=20 oraz liczby grup równej liczbie osób wynoszącej 8.



Rysunek 5: Twarze zredukowane J=20 (HOSVD)

Na rysunku 6 przedstawiono wynik grupowania metodą k-średnich dla J=30 oraz liczby grup równej liczbie osób wynoszącej 8.



Rysunek 6: 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.

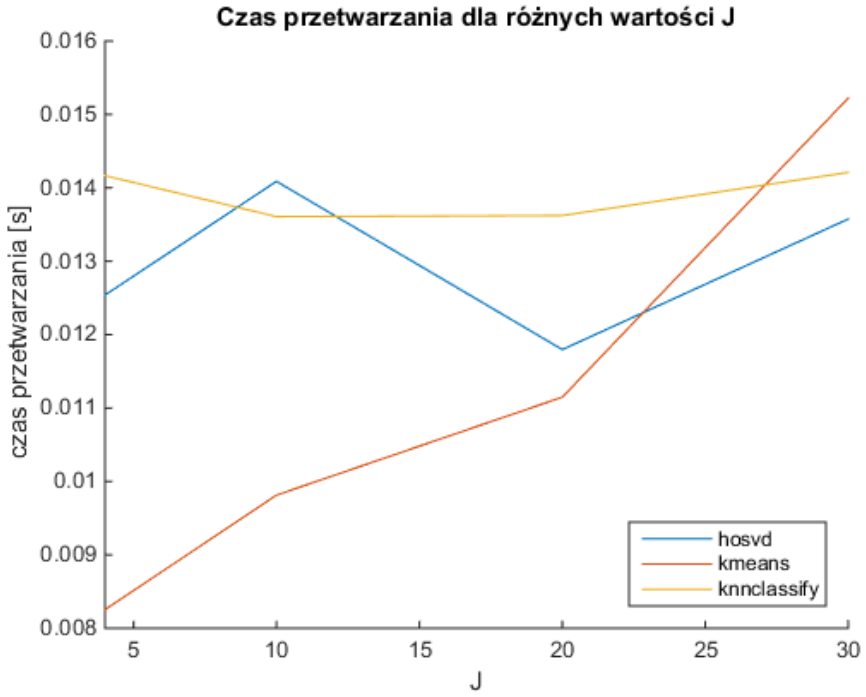
Tabela 2: Otrzymane metryki dla różnych wartości parametru J (k-najbliższych sąsiadów)

	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 7 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 7: Czas przetwarzania w zależności od wartości J

3 Podsumowanie

Podczas prac nad zadaniami zapoznano się z metodami redukcji wymiarowości przy pomocy nieujemnej faktoryzacji macierzy i dekompozycji tensorów.

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