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

Wygenerować faktory $A = [a_{ij}] \in R_+^{IxJ}$ i $X = [x_{jt}] \in R_+^{JxT}$, gdzie $a_{ij} = max(0, \breve{a}_{ij})$ i $x_{jt} = max(0, \breve{x}_{jt})$ oraz $\breve{a}_{ij}, \breve{x}_{jt} \sim N(0,1)$ (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 Algorytm ALS

Algorytm ALS polega na N-krotnym powtórzeniu pętli, której zadaniem jest obliczenie tensorów A oraz X.

W pierwszym etapie tensory A i X inicjalizowane są liczbami losowymi zależnymi od rzędu faktoryzacji. Następnie następuje N-krotne powtórzenie pętli, w której estymowane są faktory stosując wykorzystując pochodną kwadratu odległości euklidesowej.
$\nabla_A D(Y|AX) = (AX - Y)X^T = 0 \Rightarrow AXX^T = YX^T \Rightarrow A = YX^T(XX^T)^{-1}$
$\nabla_X D(Y|AX) = A^T(AX - Y) = 0 \Rightarrow A^TAX = A^TY \Rightarrow X = (A^TA)^{-1}A^TY$
Wykorzystując otrzymane wyniki obliczany jest błąd residualny.

1.2 Algorytm MUE

1.3 Algorytm HALS

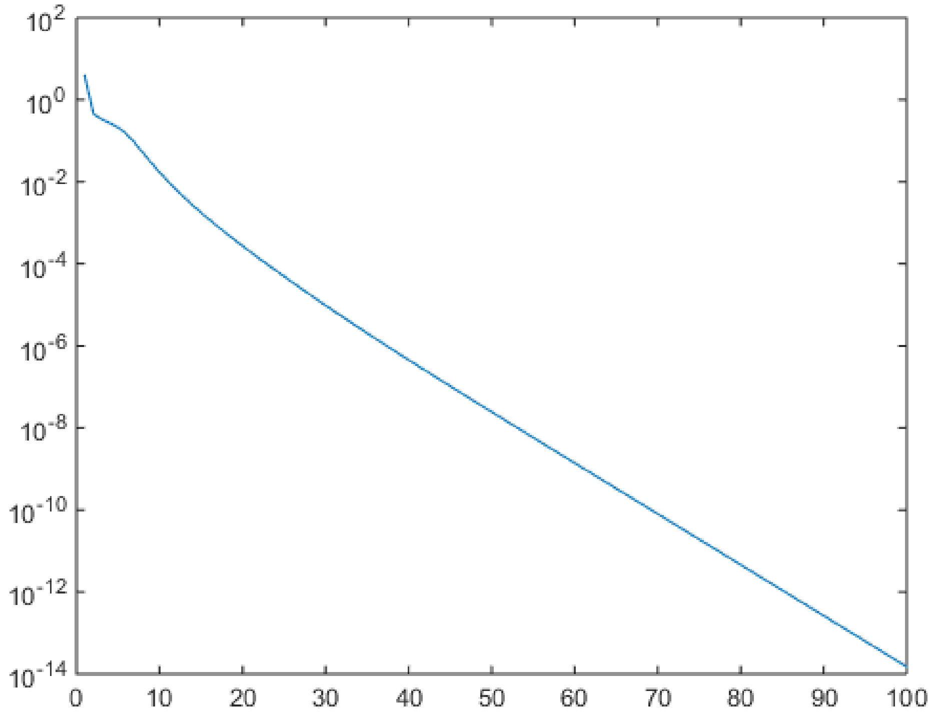
1.4 Realizacja

Listing 1: Skrypt z realizacją w środowisku MATLAB

```
clear

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

Y = Aw*Xw;

% algorytm optymalizacji naprzemiennej (als)

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

MaxIter = 100;

for k = 1:MaxIter

    A = max(0,Y*X'*inv(X*X'));
    A = A*diag(1./sum(A,1));
```

```
    X = max(0,inv(A'*A)*A'*Y);
    res(k) = norm(Y - A*X,'fro')/norm(Y,'fro');

end

figure
semilogy(res)

% to samo ale z wykorzystaniem wbudowanej funkcji

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);

% ---- to co pisal na konsoli ----

Aws = Aw*diag(1./sum(Aw,1));
grid on
eps
Aws(1:5,:)
```

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 $Y = \in R^{I_1 x I_2 x I_3}$, 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).

3.1 Opis metody

3.2 Algorytm

3.3 Realizacja

Rysunek 2: Twarze wykorzystane podczas testów

Listing 2: 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_rands = zeros(1,length(J_serie));
res_time_all = zeros(1,length(J_serie));
res_time_train = zeros(1,length(J_serie));
res_acc = 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_all(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)
    kmeans_result = kmeans(C, Persons);
    res_groups_kmeans = [res_groups_kmeans kmeans_result];
    [res_acc(J_current), res_rands(J_current), ~] = AccMeasure(P,
        kmeans_result');

    % dekompozycja hosvd
    tic
    [Ar, Br, Cr, Gr, Yr_hat] = skrypt_zad3_hosvd(Y_train, J);
    res_time_train(J_current) = toc;

    % 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)
    mdl_class = fitcknn(Cr,Class_train_idx,'NumNeighbors',1);
    prediction = predict(mdl_class, Ct);

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

end
```

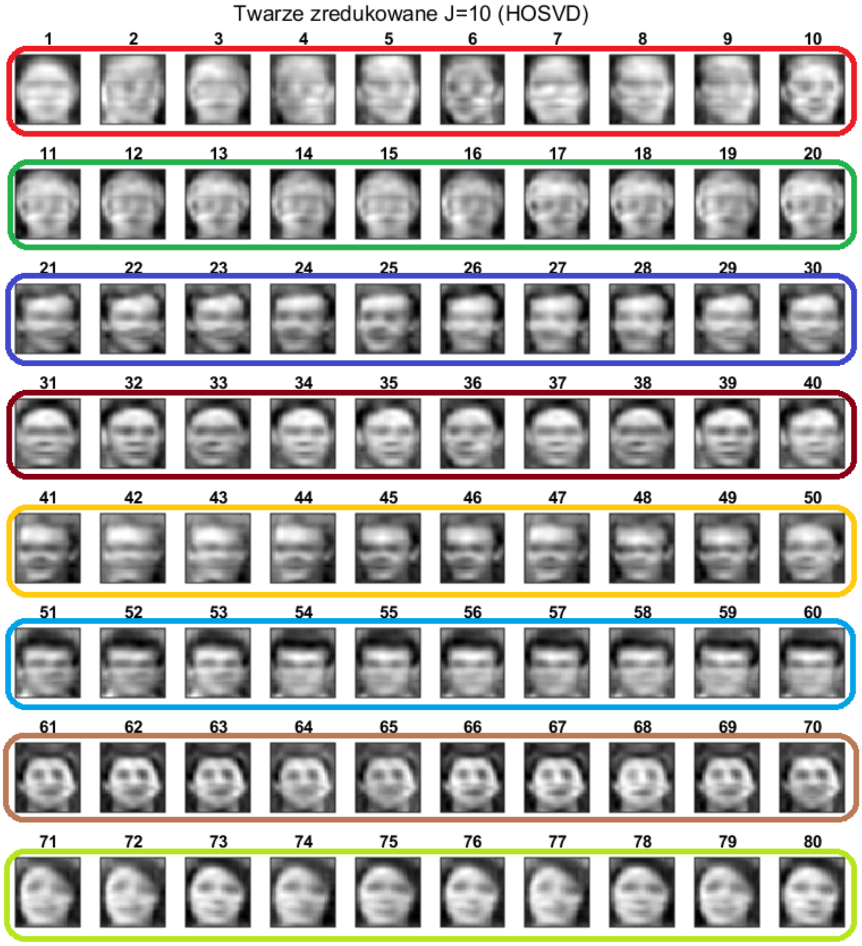
Listing 3: Funkcja realizująca algorytm HOSVD

```
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
```

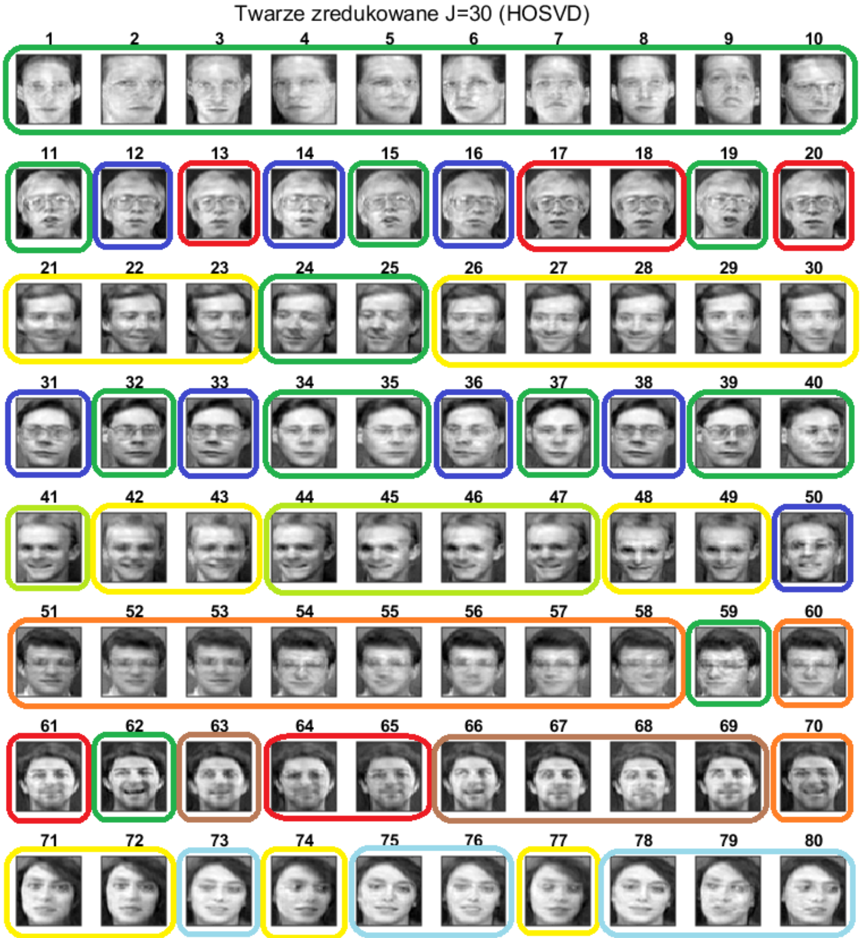
3.4 Wyniki

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 $\widehat{U}^{(3)}$ 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/`