

POLITECHNIKA WROCŁAWSKA

INTELIGENCJA OBLICZENIOWA I JEJ ZASTOSOWANIA

Ćwiczenie 1
Metody redukcji wymiarowości
Analiza składowych głównych

Autorzy:

Paweł ANDZIUL 200648

Robert CHOJNACKI 200685

Marcin SŁOWIŃSKI 200638

Prowadzący:

dr hab. inż. Rafał ZDUNEK

7 czerwca 2017

Spis treści

1	Zadanie 1	2
1.1	Opis metody	2
1.2	Algorytm	2
1.3	Realizacja	3
1.4	Wyniki	4
2	Zadanie 2	5
2.1	Wczytanie obrazów twarzy	5
2.2	Wyznaczenie cech holistycznych (twarzy własnych)	6
2.3	Grupowanie metodą k-średnich	9
2.3.1	Rezultaty grupowania dla obrazów zredukowanych	12
2.4	Klasyfikacja za pomocą klasyfikatora k-NN	15
3	Zadanie 3	17
3.1	Algorytm Powera	17
3.2	Algorytm Lanczosa	17
3.3	Wyniki	17
4	Podsumowanie	17

1 Zadanie 1

Dla danych:

$$X = [2.50.52.21.93.12.3211.51.1; 2.40.72.92.232.71.61.11.60.9];$$

- a. Zaimplementować metodę PCA w Matlabie. Do wyznaczenia par własnych macierzy kowariancji można zastosować wbudowaną funkcję `eig(.)` lub `eigs(.)`.
- b. Wyznaczyć składowe główne i wektory cech.
- c. Pokazać na rysunku punkty obserwacji oraz wyznaczone wielkości.

1.1 Opis metody

Metoda PCA (ang. *Principal Component Analysis*) jest jedną ze statystycznych metod analizy czynnikowej, która pozwala na odnajdywanie pewnych struktur w zbiorze zmiennych losowych. Analiza składowych głównych oparta jest na wykorzystaniu pojęć ze statystyki, jakimi są m.in. korelacja i wariancja. Wielowymiarowe dane z reguły nie są równomiernie rozrzucone wzdłuż wszystkich kierunków układu współrzędnych, ale koncentrują się w pewnych podprzestrzeniach oryginalnej przestrzeni. Celem PCA jest znalezienie tych podprzestrzeni w postaci składowych głównych, tzn. kierunków przy których wartość wariancji (lub korelacji) jest zmaksymalizowana. Analiza może być oparta albo na macierzy korelacji, albo macierzy kowariancji utworzonych ze zbioru wejściowego.

Metoda PCA jest zazwyczaj używana do redukcji rozmiaru danych poprzez odrzucenie ostatnich czynników, tzn. takich, których wariancja jest najniższa. Oznacza to, że n -wymiarowy zbiór danych możemy ograniczyć wyznaczając n -wektorów własnych, z których wybieramy tylko k -wektorów, tak aby $k \leq n$. Zrzutowanie danych na przestrzeń rozpiętą przez k -wybranych wektorów pozwala zredukować wymiarowość danych.

1.2 Algorytm

Jako dane wejściowe podawana jest macierz zawierająca kolejne obserwacje, na podstawie których będą wyznaczane główne składowe. Algorytm PCA składa się z następujących kroków:

1. Wyznaczenie wartości średnich dla wierszy
2. Odjęcie od macierzy wejściowej wyliczonych poprzednio średnich
3. Wyznaczenie macierzy kowariancji
4. Wyznaczenie składowych głównych
5. Wybór najlepszych składowych
6. Rzutowanie na wektory własne

Na początku należy przeprowadzić normalizację. W ten sposób zapewniamy, że cechy szerzej rozłożone nie zdominują cech mocniej skoncentrowanych. Normalizacja polega na wyznaczeniu średnich wartości kolejnych cech dla wszystkich obserwacji. Następnie od macierzy wejściowej odejmujemy wcześniej wyliczone średnie - od każdego elementu macierzy odejmujemy średnią dla wiersza, w którym się znajduje. Dla tak uzyskanych danych należy policzyć macierz kowariancji.

Kolejnym krokiem jest wyznaczenie wektorów i wartości własnych. Wylicza się k -ortonormalnych wektorów, które tworzą bazę dla unormowanych danych wejściowych. Są to wektory jednostkowe, wskazujące w kierunku prostopadłym do pozostałych wektorów z utworzonej bazy. Liczba wektorów własnych jest związana z wymiarem rozpatrywanych danych, a wartości własne określają długość wektora.

Następnie porządkujemy wartości własne w kolejności malejącej. Redukcja opiera się na wyborze wektorów własnych z największą wartością własną. Można powiedzieć, że maksymalizujemy zmienność danych wraz z jednoczesną minimalizacją ubytku informacji spowodowanej ich redukcją. Redukcja dokonuje się poprzez mnożenie macierzy wybranych wektorów przez n -wymiarową macierz danych.

Jeżeli wybierzemy mniejszą liczbę wektorów własnych (k z n) otrzymujemy zredukowaną przestrzeń danych. Prowadzi to do utraty części oryginalnej informacji, a rozmiar straty zależy od doboru wektorów. Jeżeli wybierzemy wszystkie wektory własne, uzyskujemy jedynie obrót macierzy danych wejściowych, bez utraty żadnych informacji.

1.3 Realizacja

Opracowano skrypt w środowisku Matlab realizujący metodę PCA zgodnie z algorytmem opisanym wcześniej. Wykorzystano wbudowane funkcję do obliczenia i odjęcia wartości średnich (`mean()`, `bsxfun()`) oraz do wyznaczenia macierzy kowariancji (`cov()`). W celu obliczenia par własnych macierzy kowariancji zastosowana została metoda `eigs()`, w której pierwszym argumentem jest macierz kowariancji, a drugim ilość wektorów własnych.

Listing 1: Wyznaczanie wartości własnych i wektorów własnych

```

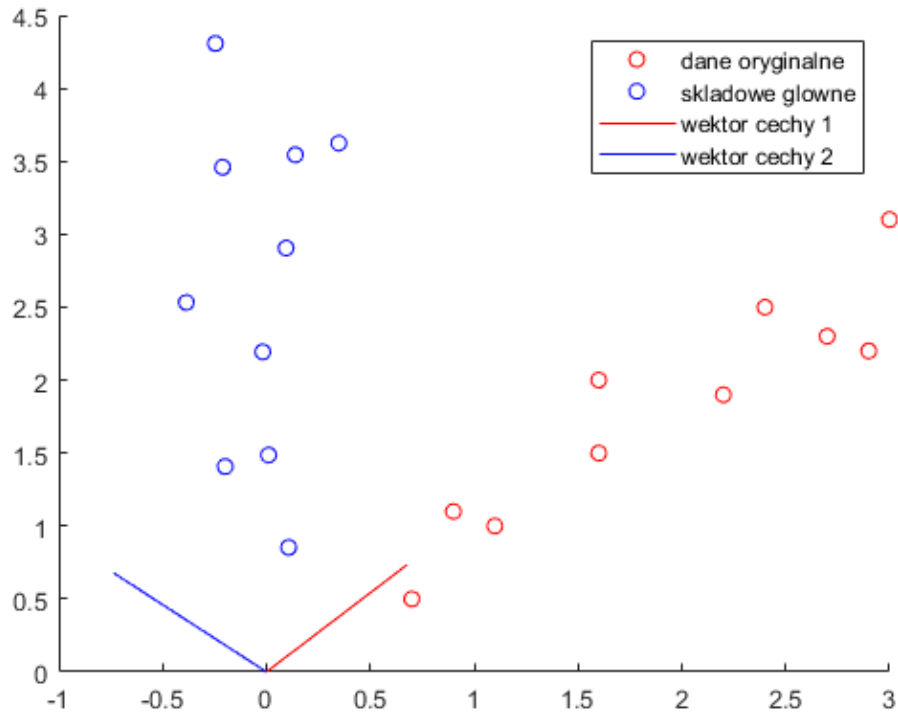
1 % dane wejsciowe
2 x = [2.5 0.5 2.2 1.9 3.1 2.3 2 1 1.5 1.1;
3       2.4 0.7 2.9 2.2 3 2.7 1.6 1.1 1.6 0.9];
4 % obliczenie wartosci srednich
5 % odjecie ich od macierzy
6 x_norm = bsxfun(@minus, x, mean(x, 2));
7 % wyznaczenie macierzy kowariancji
8 s = cov(x_norm');
9 % wyznaczenie wartosci i wektorow wlasnych
10 [eigenvectors, eigenvalues] = eigs(s, 2);
11 % porzadkujemy wartosci wlasne w kolejnosci malejacej
12 [eigenvalues order] = sort(diag(eigenvalues), 'descend');
13 eigenvectors = eigenvectors(:,order);
14 % rzutowanie
15 pcs = eigenvectors' * x;
16
17 figure;
18 hold on
19 plot(x(2,:), x(1,:), 'or', pcs(2,:), pcs(1,:), 'ob')
20 plotv(eigenvectors(:,1), '-r');
21 plotv(eigenvectors(:,2), '-b');
22 legend('dane oryginalne','skladowe glowne','wektor cechy 1','wektor cechy
23         2','Location','northeast','Orientation','vertical')
24 hold off

```

Pozostałe funkcje służą do wykreślenia wykresu zawierającego kolejno: dane wejściowe, składowe główne oraz wyznaczone wektory własne.

1.4 Wyniki

Dane wejściowe, składowe główne oraz wektory własne przedstawiono na rysunku 1. Wyznaczone wektory i wartości własne przedstawiono w tabelach 1 oraz 2. Można zauważyć, że zgodnie z założeniami metody PCA, wektory własne są ortogonalne. Jeden z wektorów pokazuje kierunek, w którym wariancja jest największa.



Rysunek 1: Ilustracja punktów obserwacji i składowych głównych

Tabela 1: Wyznaczone wektory własne.

	x	y
wektor cechy 1	0.7352	0.6779
wektor cechy 2	0.6779	-0.7352

Tabela 2: Wyznaczone wartości własne.

wektor cechy 1	1.2840
wektor cechy 2	0.0491

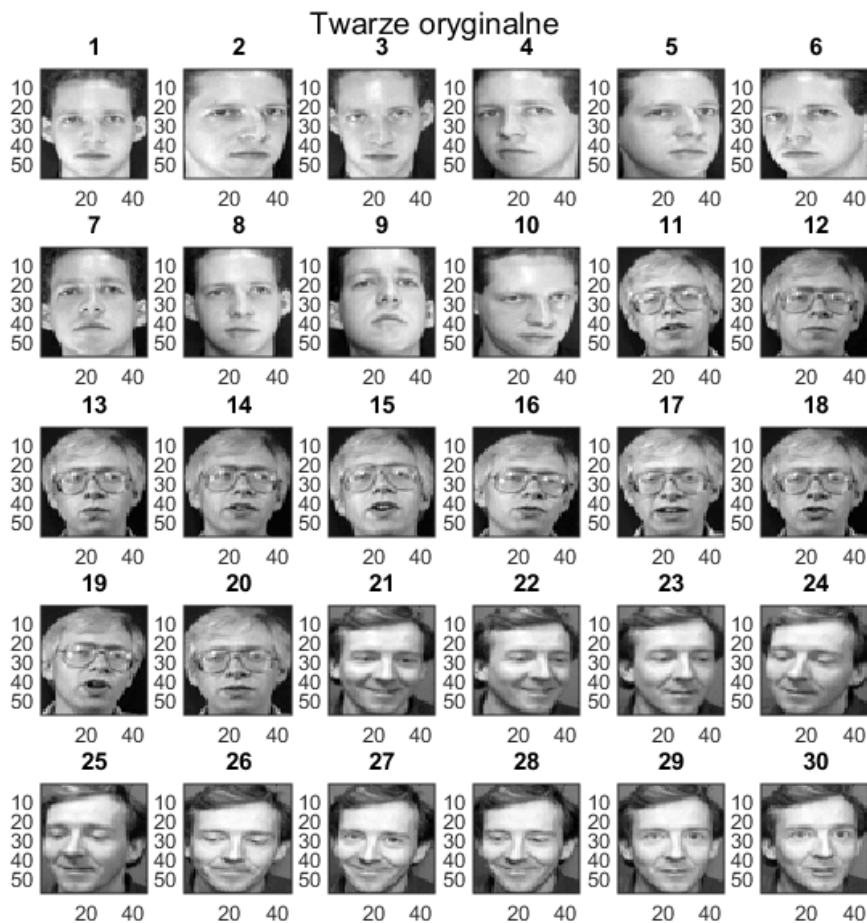
Na wykresie można zauważyć poprawne działanie funkcji PCA. Dane rozkładają się wzdłuż oznaczonych wektorów.

2 Zadanie 2

Dla obrazów twarzy z bazy ORL (lub podobnej) wyznaczyć cechy holistyczne (twarze własne) dla różnej liczby estymowanych komponentów głównych ($J = 4, 10, 20, 30$). Pogrupować obrazy stosując metodę k -średnich, do obrazów oryginalnych oraz zredukowanych. Badania przeprowadzić dla różnej liczby grup. Porównać dokładność i czas grupowania. Następnie dokonać klasyfikacji obrazów w obu przestrzeniach (oryginalnej i zredukowanej) za pomocą klasyfikatora k -NN. Porównać efekty klasyfikacji z efektami grupowania.

2.1 Wczytanie obrazów twarzy

Na ilustracji 2 zamieszczono wykorzystane dane twarzy 3 różnych osób. Oryginalne twarze pochodzą ze strony [6]. Zdjęcia twarzy są już znormalizowane.



Rysunek 2: Obrazy twarzy wykorzystane do obliczeń

Listing 2: Wczytywanie danych twarzy do macierzy

```
1 clc;  
2 clear;
```

```

3 close all;
4
5 load('FaceData_56_46.mat');
6
7 Persons = 3;
8 ImagesPerPerson = 10;
9
10 %wczytywanie danych
11 Group = [];
12 M = [];
13 for p=(1:Persons)
14     for i=(1:ImagesPerPerson)
15         x = FaceData(p, i).Image;
16         [irow, icol] = size(x);
17         x = double(x);
18         temp = reshape(x', irow * icol, 1);
19         M = [M temp];
20         Group = [Group p];
21     end
22 end
23
24 figure;
25 suptitle('Twarze oryginalne');

```

2.2 Wyznaczenie cech holistycznych (twarzy własnych)

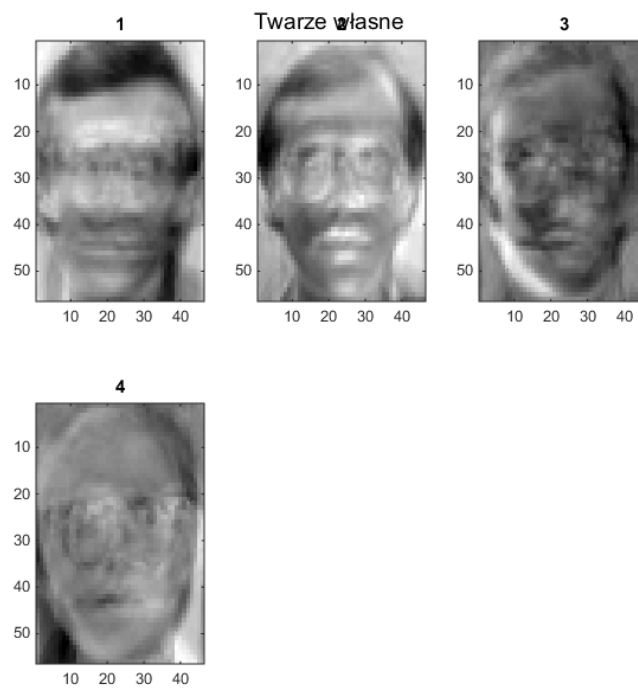
Do wyznaczania twarzy własnych zastosowano podobnie jak w zadaniu poprzednim metodę PCA. Na listingu 3 fragment odpowiedzialny za ten etap.

Listing 3: Wyznaczanie twarzy własnych

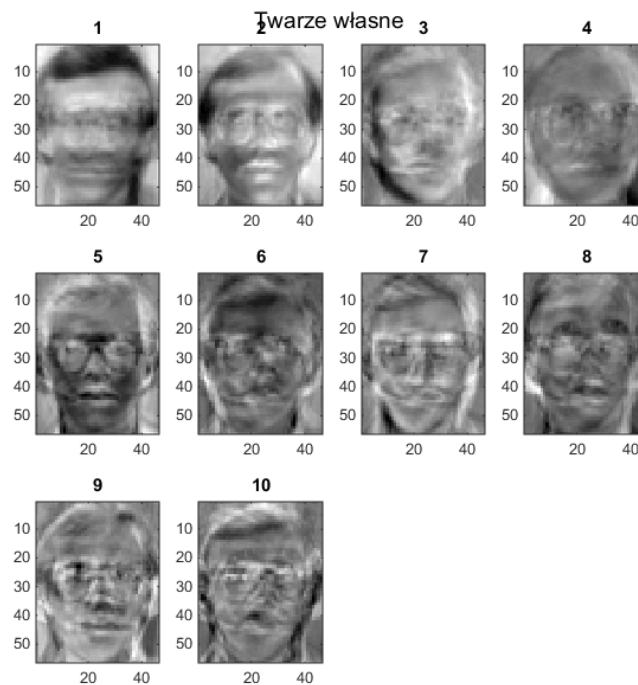
```

1 suptitle('Twarze oryginalne');
2 nOfImages = Persons*ImagesPerPerson;
3 for i=(1:nOfImages)
4     C = M(:,i);
5     CC = reshape(C, [46, 56]);
6     subplot(round(sqrt(nOfImages)), round(sqrt(nOfImages)) + 1, i);
7     imagesc(CC');
8     title(i)
9     colormap gray;
10 end
11
12 for group_size=(1:3)
13     tic;
14
15     % ksrednich dla obrazow oryginalnych
16     kmeans_result_orig = kmeans(M', group_size);

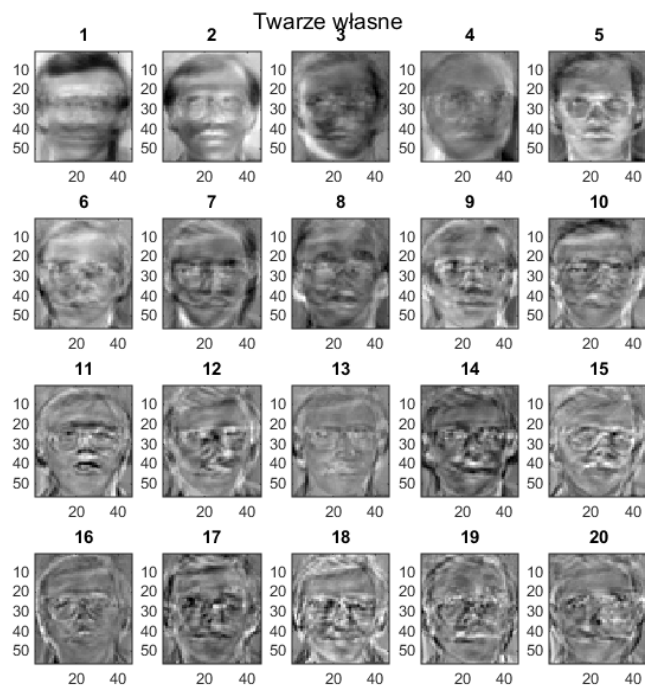
```



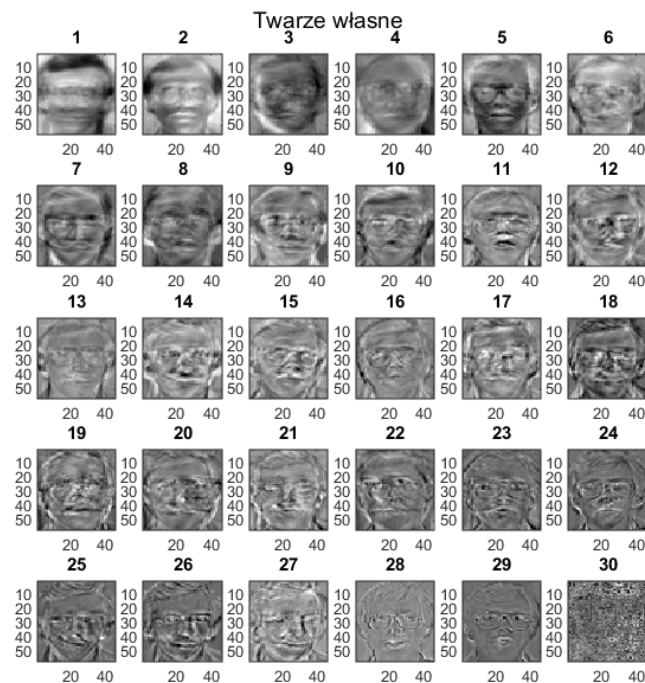
Rysunek 3: Wyznaczone twarze własne dla $J=4$



Rysunek 4: Wyznaczone twarze własne dla $J=10$



Rysunek 5: Wyznaczone twarze własne dla $J=20$



Rysunek 6: Wyznaczone twarze własne dla $J=30$

2.3 Grupowanie metodą k-średnich

Listing 4: Grupowanie twarzy oryginalnych i własnych

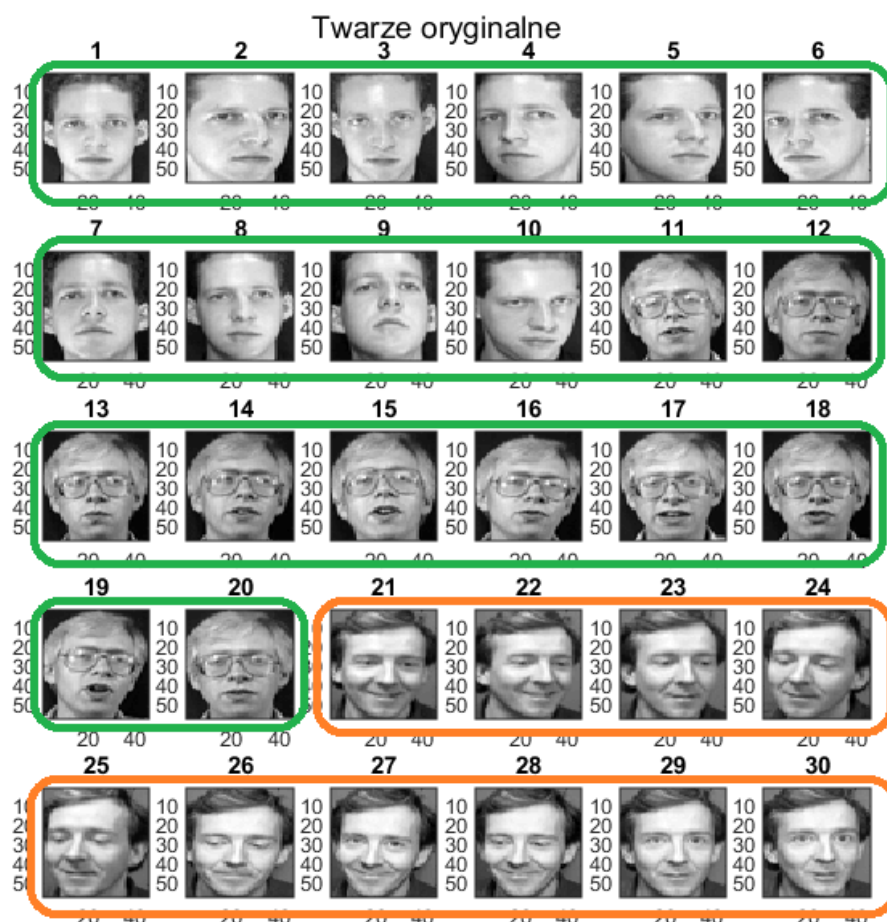
```
1      CC = reshape(C, [46, 56]);
2      subplot(round(sqrt(J)), round(sqrt(J)) + 1, i);
3      imagesc(CC');
4      title(i)
5      colormap gray;
6  end
7
8      known_groups = ones(1,J);
9
10     tic;
11
12     % ksrednich dla obrazow redukowanych
13     kmeans_result_eigen = kmeans(Z', Persons);
14     [acc_eigen, rand_index_eigen, match_eigen] = AccMeasure(known_groups,
15         kmeans_result_eigen');
16
17     disp(sprintf('Czas grupowania dla obrazow redukowanych J=%d: %2.3fs',J, toc))
18     disp(sprintf('Dokladnosc grupowania dla obrazow redukowanych J=%d: %2.2f',J,
19         acc_eigen))
```

Wyniki grupowania dla obrazów oryginalnych dla różnych rozmiarów grup przedstawiono w tabeli 3.

Tabela 3: Jakość grupowania dla różnych wielkości grup (n) dla obrazów oryginalnych

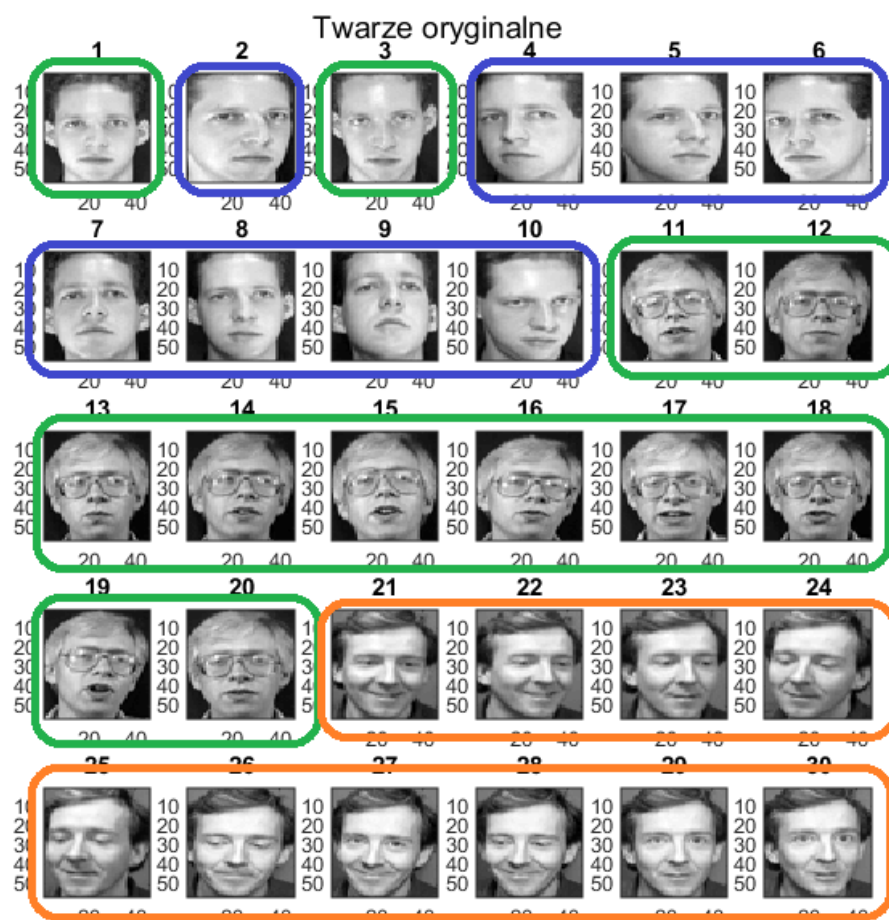
n	AccMeasure	Czas przetwarzania [s]
1	31.03	0.011
2	77.01	0.045
3	91.72	0.140

Poniżej przedstawiono ilustrację przypisania poszczególnych zdjęć do grup przez algorytm dla 2 oraz 3 grup.



Rysunek 7: Przydzielone grupy dla poszczególnych twarzy dla zdjęć oryginalnych przy 2 grupach

W tym przypadku algorytm poradził sobie najlepiej jak potrafił. Poprawnie przyporządkował dwie pierwsze osoby do pierwszej grupy i trzecią do grupy drugiej. Nierówne przyporządkowanie zdjęć do innych osób oznaczałoby popełnienie błędu.



Rysunek 8: Przydzielone grupy dla poszczególnych twarzy dla zdjęć oryginalnych przy 3 grupach

Jak widać na powyższej ilustracji8 algorytm nie poradził sobie w pełni z rozpoznaniem twarzy pierwszej osoby. Zdjęcia 1 i 3 włączone zostały do grupy osoby drugiej.

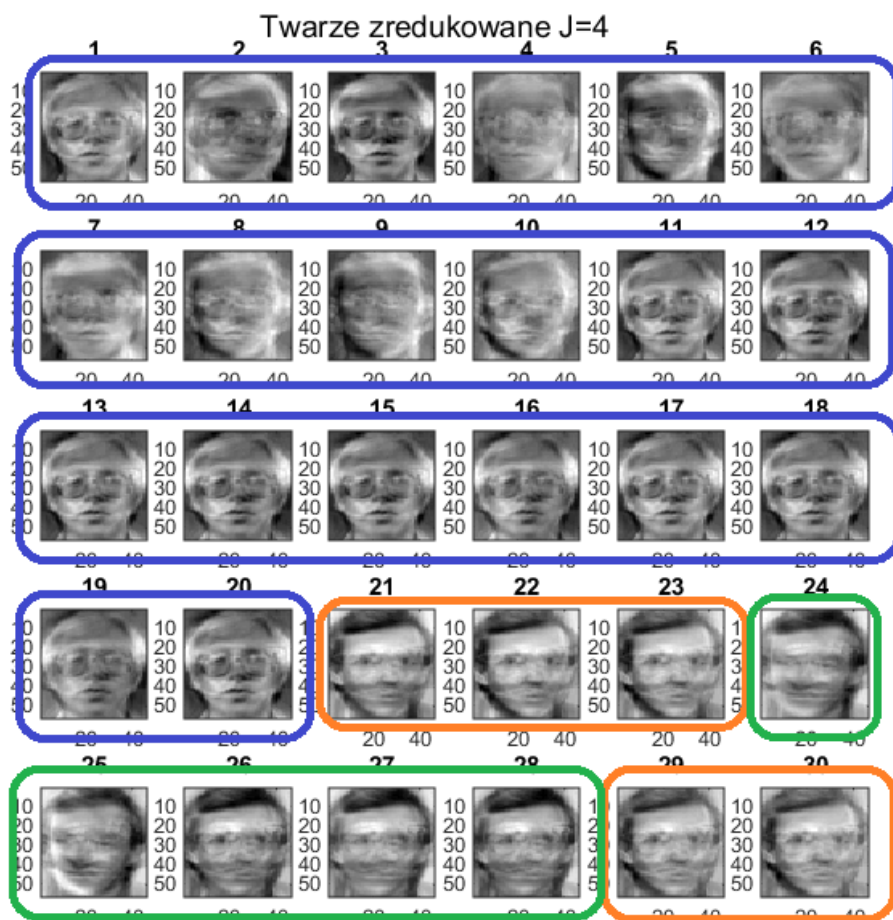
Wyniki te odpowiadają skuteczności zamieszczonym w tabeli3.

Wyniki grupowania dla twarzy własnych dla różnych wartości J przedstawiono w tabeli4.

Tabela 4: Jakość grupowania dla różnych wartości J dla twarzy redukowanych

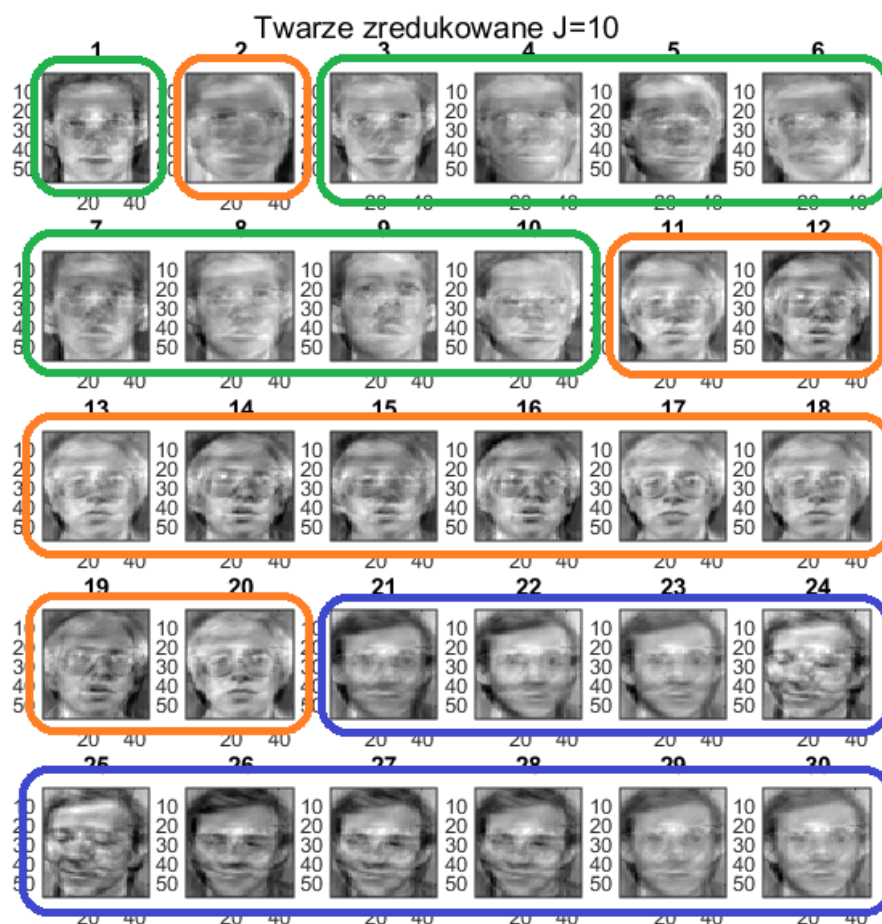
J	AccMeasure	Czas przetwarzania [s]
4	90.00	0.004
10	96.67	0.004
20	93.33	0.003
30	93.33	0.003

2.3.1 Rezultaty grupowania dla obrazów zredukowanych



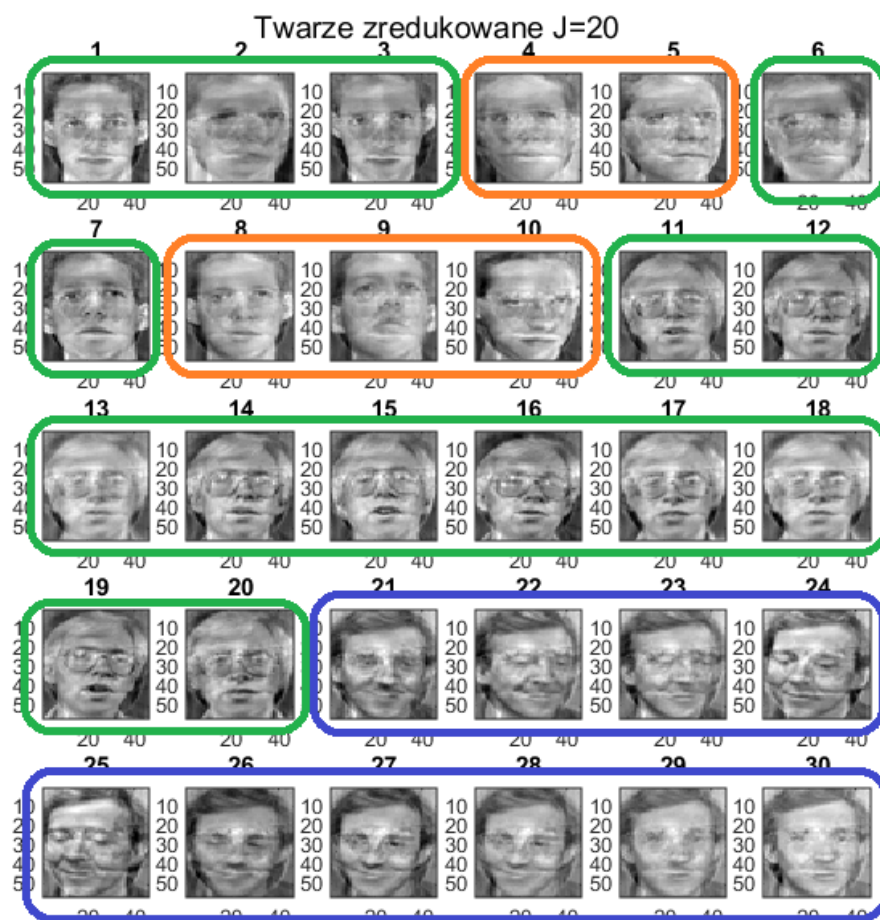
Rysunek 9: Twarze zredukowane J=4

W tym przypadku algorytm przypisywał większość twarzy do jednej osoby, aż do momentu, gdy zdjęcie znacznie się wyróżniało. Nie poradził sobie poprawnie z rozpoznaniem ostatniej z osób, gdyż jej zdjęcia zostały przypisane do dwóch różnych grup.



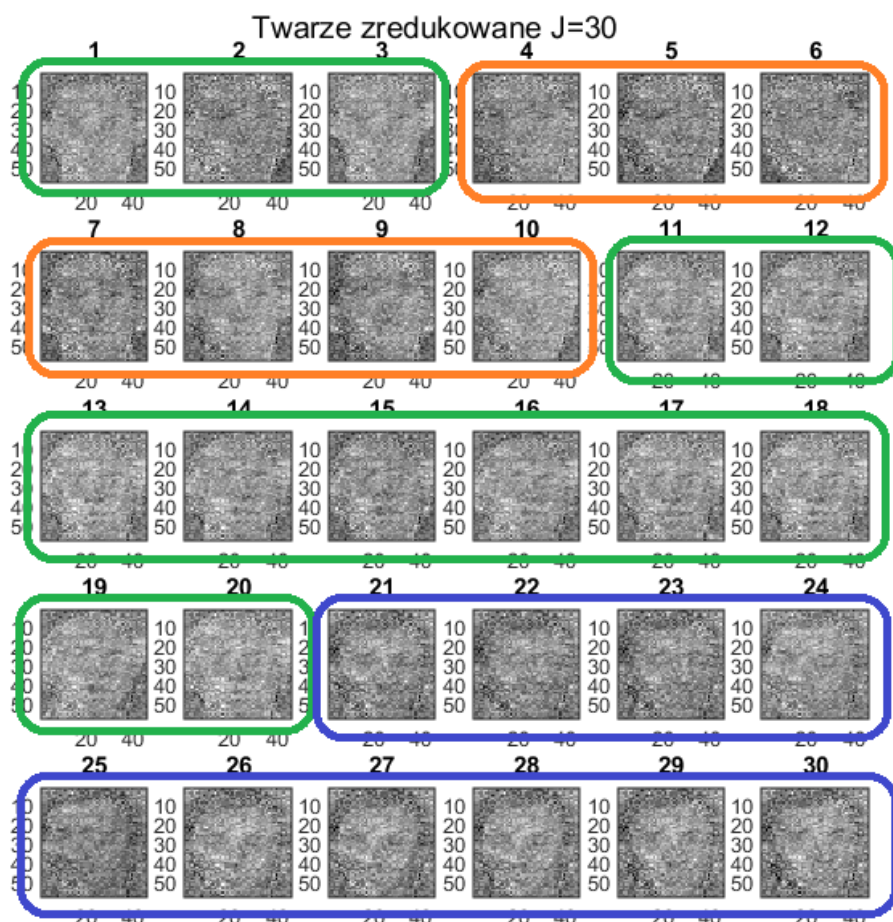
Rysunek 10: Twarze zredukowane J=10

Dla $J=10$ algorytm się okazał się skuteczniejszy niż we wcześniejszym przypadku. Zdjęcia pogrupowane zostały prawie idealnie. Tylko jedno ze zdjęć zostało błędnie przypisane.



Rysunek 11: Twarze zredukowane J=20

W tym przypadku algorytm popełnił więcej błędów, niepoprawnie klasyfikując zdjęcia pierwszej osoby. Zdjęcia trzeciej z osób zostały przypisane poprawnie, ponieważ wyróżniają się dużo bardziej niż pozostałe.



Rysunek 12: Twarze zredukowane J=30

Dla $J=30$ algorytm pogrupował zdjęcia popełniając ten sam błąd co w przypadku $J=20$. Zdjęcia pierwszej osoby zostały przypisane zarówno do pierwszej i drugiej grupy. Mimo dużej utraty jakości zdjęcia grupa trzecia została przypisana zgodnie z rzeczywistością.

2.4 Klasyfikacja za pomocą klasyfikatora k-NN

Zgodnie z dokumentacją metody `knnclassify` [3] umożliwia ona klasyfikację pewnej próbki danych na podstawie danych dla których klasyfikacja jest znana.

$$Class = knnclassify(Sample, Training, Group) \quad (1)$$

, parametry *Training* oraz *Group* zawierają informację o podziale danych na grupy natomiast *Sample* jest zbiorem którego klasyfikację pragniemy przeprowadzić. Wynikiem jest *Class* będący tego samego typu co *Group* tzn. dziedzina klasyfikacji jest taka sama.

Poniżej (listing ??) zamieszczono implementację klasyfikacji.

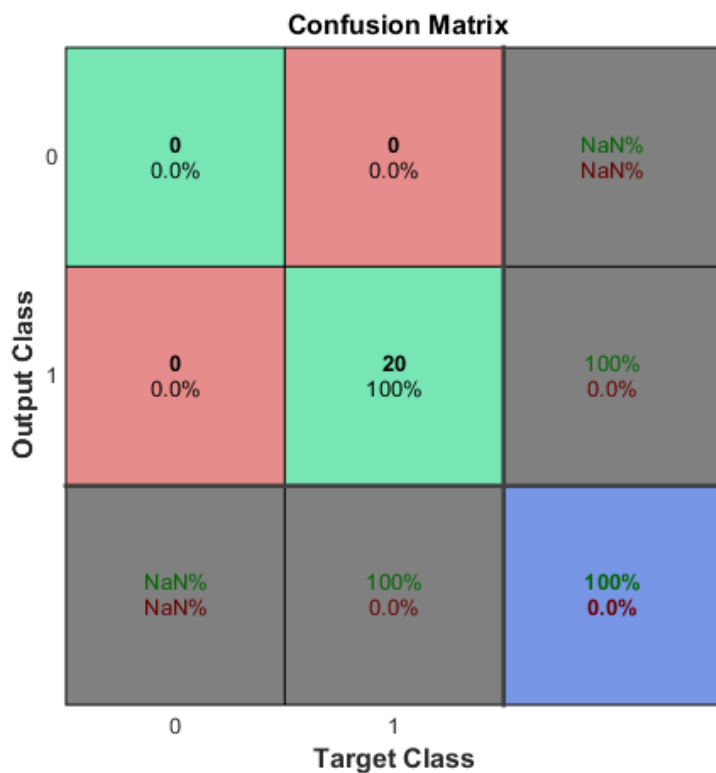
Listing 5: Fragment skryptu w Matlabie obejmujący klasyfikację k-NN

1
2


```

3 Class = knnclassify(V', M', Group);
4 [acc_classify, rand_index_classify, match_classify] =
    AccMeasure(known_groups, Class');
5
6 figure;
7 plotconfusion(known_groups, Class')
8
9 end

```



Rysunek 13: Confusion matrix dla J=20

Wyniki grupowania dla twarzy własnych dla różnych wartości J przedstawiono w tabeli 5.

Tabela 5: Jakość klasyfikacji dla różnych wartości J dla twarzy redukowanych

J	AccMeasure	Czas przetwarzania [s]
4	66.67	0.005
10	66.67	0.005
20	66.67	0.005
30	73.33	0.007

3 Zadanie 3

Wyznaczyć pary własne macierzy kowariancji za pomocą algorytmów: Powera oraz Lanczosa. Zaimplementować algorytmy i zastosować je do rozwiązywania powyższych zadań. Porównać wyniki.

3.1 Algorytm Powera

Listing 6: Fragment skryptu w Matlabie

```
1 function [new_result] = Power_f(X, tolerance)
2
3     n = size(X);
4     n = n(1);
5
6     new_result = ones(n,1);
7     m = 0;
8
9     while(1)
10         m_old = m;
11         old_result = new_result;
12         new_result = X * new_result;
13
14         m = max(new_result);
15         new_result = new_result/m;
16
17         if abs(m-m_old) < tolerance && norm(new_result-old_result,2) < tolerance
18             break;
19         end
20     end
21 end
```

Algorytm Powera polega na kolejnym przemnażaniu macierzy wejściowej przez wektor wynikowy do momentu, gdy wektor uzyskany w wyniku działania nie różni się bardziej od wynikowego o więcej niż założona wartość tolerancji.

Powyższe założenie wynika z faktu, że kolejne wektory dążą do pewnej zwielokrotnionej wartości. Ważnym jest by tę wartość wychwycić nie wykonując nadmiarowych obliczeń.

W zaimplementowanej funkcji przechowywane są dwa wektory - stary i nowy. Wektory te odejmowane są od siebie i normalizowane. Jeżeli różnica między nimi jest mniejsza niż założona tolerancja zwracany jest wektor nowszy jako bliższy idealnemu rozwiązaniu.

3.2 Algorytm Lanczosa

3.3 Wyniki

4 Podsumowanie

Podczas zajęć laboratoryjnych zapoznano się z metodą PCA. Poznano również podstawy grupowania i klasyfikacji danych z wykorzystaniem pakietu Matlab.

Literatura

- [1] <https://www.mathworks.com/help/nnet/ref/plotconfusion.html>
- [2] <https://www.mathworks.com/help/stats/confusionmat.html>
- [3] <https://www.mathworks.com/help/bioinfo/ref/knnclassify.html>
- [4] <http://www.kmg.zut.edu.pl/opt/wyklad/bezgrad/powell.html>
- [5] https://en.wikipedia.org/wiki/Lanczos_algorithm
- [6] <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>
- [7] Berk Gokberk, „Assignment 2: Face Recognition using PCA”, http://www2.cmpe.boun.edu.tr/courses/cmpe58Z/spring2010/files/assignment2_new.pdf