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

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 $k \leqslant n$. 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 wejsciowych. 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

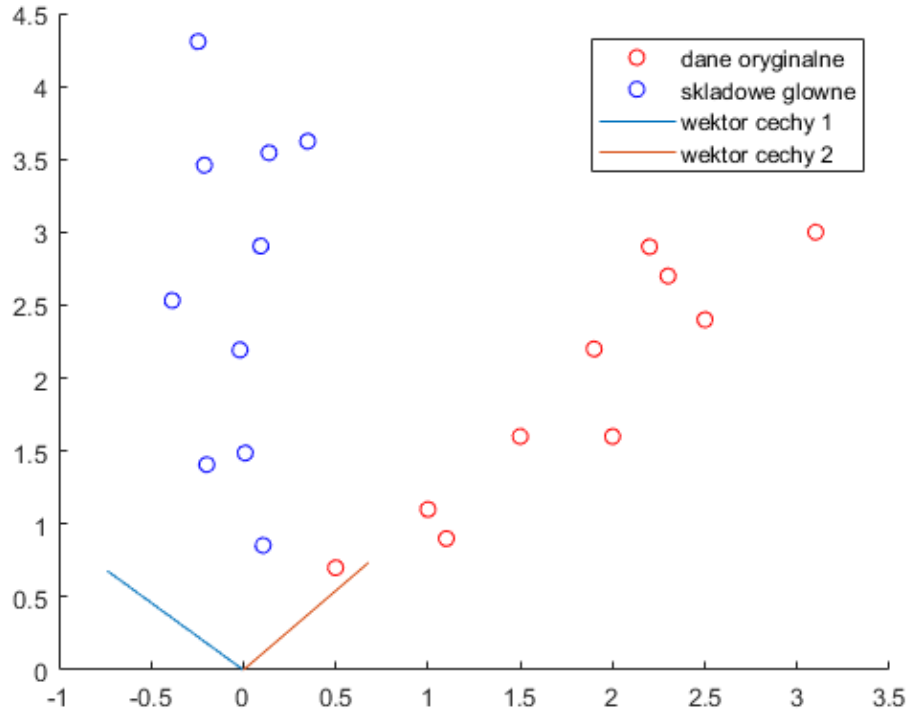
Opracowano skrypt w środowisku Matlab realizujący metodę PCA zgodnie z algorytmem opisanym wcześniej. Wykorzystano wbudowane funkcję do obliczenia i odjęcia wartości średnich [mean(.), bsxfun(.)] oraz do wyznaczenia macierzy kowariancji [cov(.)]. W celu obliczenia par własnych macierzy kowariancji zastosowana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

Listing 1: Wyznaczanie wartości własnych i wektorów własnych

```
% dane wejsciowe
x = [2.5 0.5 2.2 1.9 3.1 2.3 2 1 1.5 1.1;
    2.4 0.7 2.9 2.2 3 2.7 1.6 1.1 1.6 0.9];
% obliczenie wartosci srednich
% odjecie ich od macierzy
x_norm = bsxfun(@minus, x, mean(x, 2));
% wyznaczenie macierzy kowariancji
s = cov(x_norm');
% wyznaczenie wartosci i wektorow wlasnych
[eigenvectors, eigenvalues] = eigs(s, 2);
% rzutowanie
pcs = eigenvectors' * x;

figure;
hold on
plot(x(1,:), x(2,:), 'or', pcs(1,:), pcs(2,:), 'ob')
plotv(eigenvectors(:,1), '-');
plotv(eigenvectors(:,2), '-');
legend('dane oryginalne', 'skladowe glowne', 'wektor cechy 1','wektor cechy
    2','Location','northeast','Orientation','vertical')
hold off
```

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

1.4 Wyniki



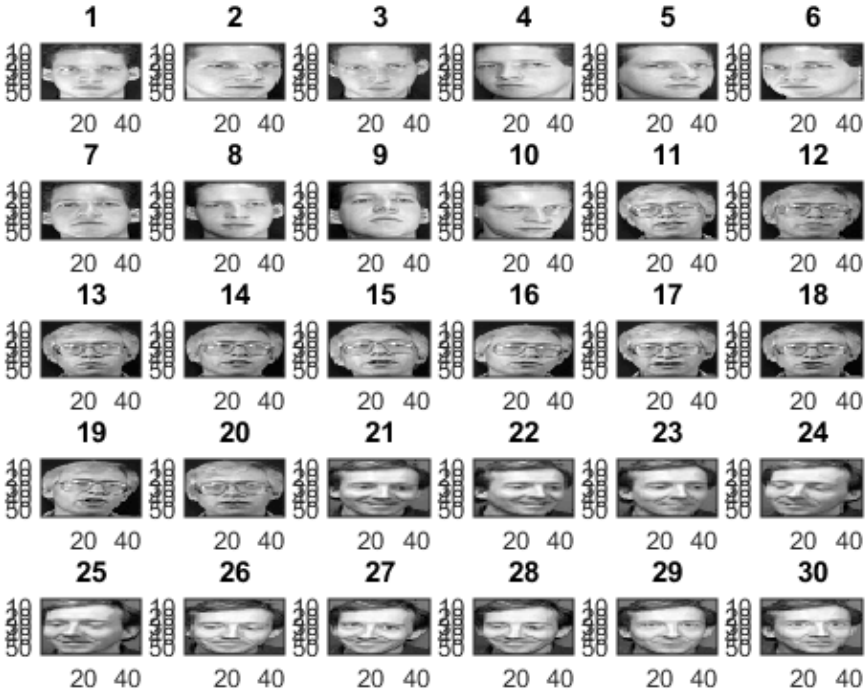
Rysunek 1: Ilustracja punktów obserwacji i składowych głównych

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

2 Zadanie 2

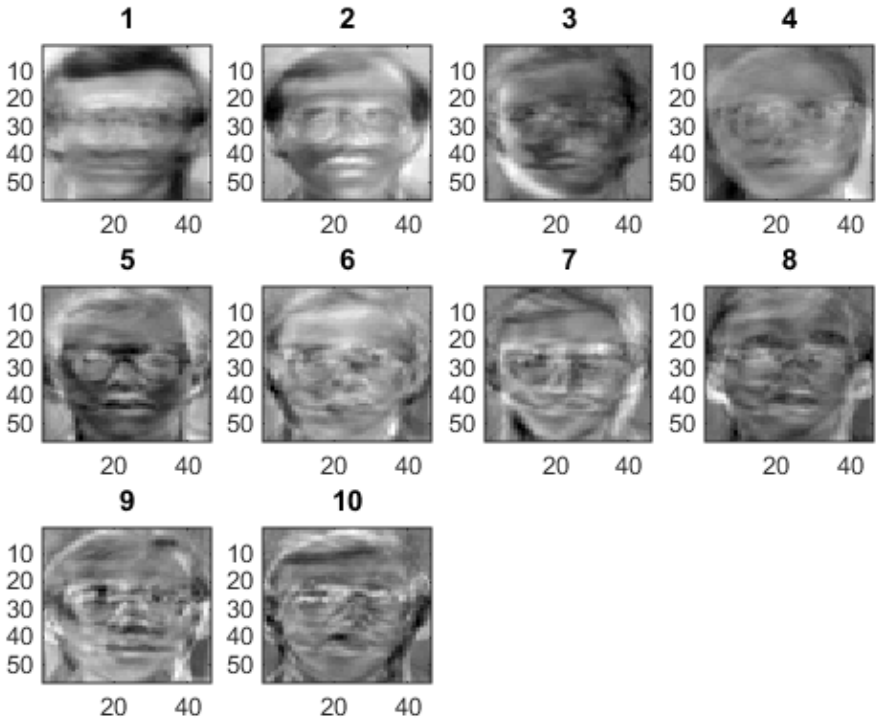
Dla obrazów twarzy z bazy ORL (lub podobnej) wyznaczyć cechy holistyczne (twarze własne) dla różnej liczby estymowanych komponentów głównych ($J = 4, 10, 20, 30$). Pogrupować obrazy stosując metodę k-średnich, do obrazów oryginalnych oraz redukowanych. Badania przeprowadzić dla różnej liczby grup. Porównać 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



Rysunek 2: Ilustracja twarzy wejściowych...

2.2 Wyznaczenie cech holistycznych (twarzy własnych)



Rysunek 3: Ilustracja ejgenfejsów...

2.3 Grupowanie metodą k-średnich

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.

$$Class = knnclassify(Sample, Training, Group) \tag{1}$$

, 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 zamieszczono..

Listing 2: Fragment skryptu w Matlabie obejmujący klasyfikację k-NN

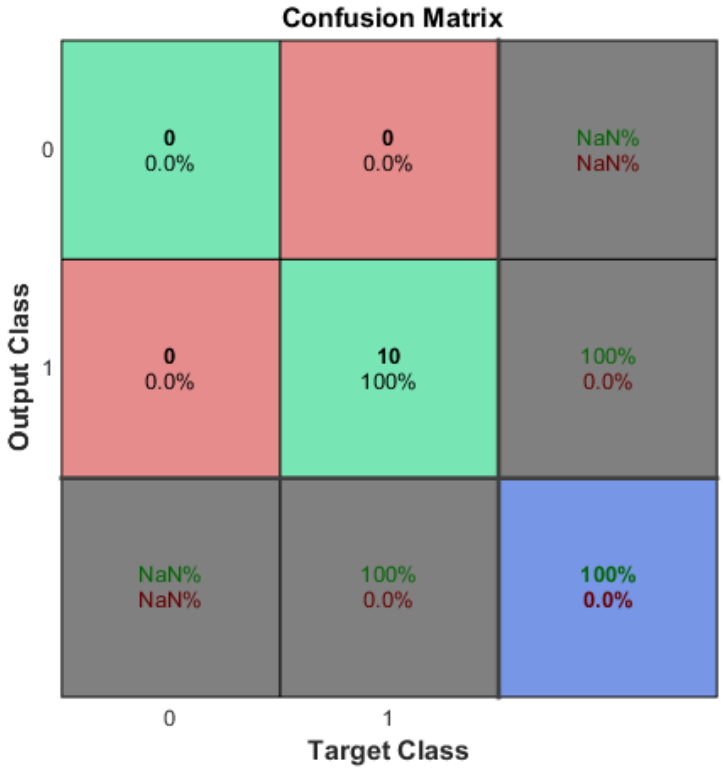
```
clc;
clear;
close all;

```

```
load('yalefaces.mat');
load('FaceData_56_46.mat');

%imagesc(yalefaces(:,:,1))

% -- FRAGMENT Z WIKIPEDII --

% J = 30;
% [h,w,n] = size(yalefaces);
% d = h*w;
% % vectorize images
% x = reshape(yalefaces,[d n]);
% x = double(x);
% %subtract mean
% x=bsxfun(@minus, x, mean(x,2));
% % calculate covariance
% s = cov(x');
% % obtain eigenvalue & eigenvector
% [V,D] = eigs(s,J);
% eigval = diag(D);
% % sort eigenvalues in descending order
% eigval = eigval(end:-1:1);
% V = fliplr(V);
% % show 0th through 15th principal eigenvectors
% eig0 = reshape(mean(x,2), [h,w]);
% figure,subplot(6,6,1)
% imagesc(eig0)
% colormap gray
% for i = 1:J
%  subplot(6,6,i+1)
%  imagesc(reshape(V(:,i),h,w))
% end


% -- NASZ KOD --

Persons = 3;
ImagesPerPerson = 10;
J = 10;

Group = [];
M = [];
for p=(1:Persons)
    for i=(1:ImagesPerPerson)
        x = FaceData(p, i).Image;
        [irow, icol] = size(x);
        x = double(x);
        temp = reshape(x', irow * icol, 1);
        M = [M temp];
        Group = [Group p];
    end
end


%[eigenvectors, eigenvalues] = eigs(M*(M'), J);
%V = eigenvectors;
```

```
61
62
63  %subtract mean
64  x=bsxfun(@minus, M, mean(M,2));
65  % calculate covariance
66  s = cov(x');
67  % obtain eigenvalue & eigenvector
68  [V,D] = eigs(s,J);
69  Z = (x') * V;
70
71
72  figure; %oryginalne
73  nOfImages = Persons*ImagesPerPerson;
74  for i=(1:nOfImages)
75      C = M(:,i);
76      CC = reshape(C, [46, 56]);
77      subplot(round(sqrt(nOfImages)), round(sqrt(nOfImages)) + 1, i);
78      imagesc(CC');
79      title(i)
80      colormap gray;
81  end
82
83  figure; %eigenfaces
84  for i=(1:J)
85      C = V(:,i);
86      CC = reshape(C, [46, 56]);
87      subplot(round(sqrt(J)), round(sqrt(J)) + 1, i);
88      imagesc(CC');
89      title(i)
90      colormap gray;
91  end
92
93  id = (1:J);
94
95  kmeans_result = kmeans(Z', Persons);
96  groups = sortrows([id', kmeans_result], 2);
97
98  known_groups = [1 2 3 2 1 2 3 2 3 1];
99
100 confusionmat_result = confusionmat(known_groups, kmeans_result);
101
102 figure;
103 plotconfusion(known_groups, kmeans_result')
104
105 [acc_eigen, rand_index_eigen, match_eigen] = AccMeasure(known_groups,
        kmeans_result);
106
107 kmeans_result_original = kmeans(M', Persons);
108 [acc_orig, rand_index_orig, match_orig] = AccMeasure(Group,
        kmeans_result_original);
109
110 Class = knnclassify(V', M', Group);
111
112 figure;
113 plotconfusion(known_groups, Class')
```

Poniżej confusion matrix..

2	0	1
3	1	0
3	0	0



Rysunek 4: Tzw. confusion matrix...

3 Zadanie 3

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

3.1 Algorytm Powera

Listing 3: Fragment skryptu w Matlabie

```
function [new_result] = Power_f(X, tolerance)

    n = size(X);
    n = n(1);

    new_result = ones(n,1);
    m = 0;

    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...

Z przeprowadzonych badań wynika...

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