

Inteligencja obliczeniowa i jej zastosowania

Laboratorium cz. IV, nr 1-2

Autorzy:

Joanna Piątek, nr indeksu: 199966

Agnieszka Wątrucka, nr indeksu: 200016

Grupa: Środa, 15:15

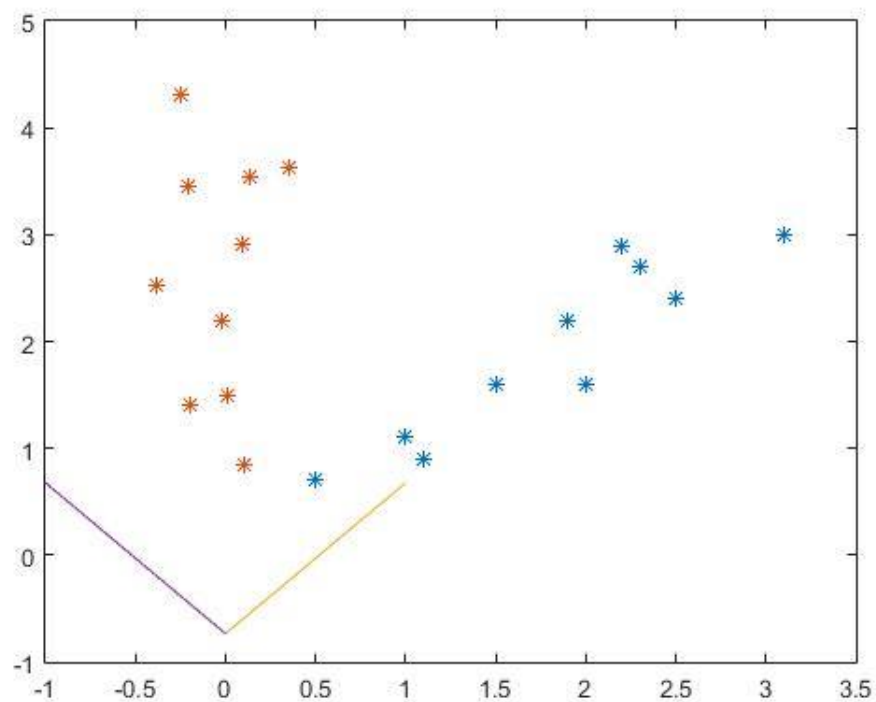
7 czerwca 2017

Prowadzący: prof. dr hab. inż. Rafał Zdunek

1 Zadanie nr 1

W ćwiczeniu nr 1 za zadanie była implementacja metody PCA w środowisku Matlab, wyznaczenie składowych głównych i wektora cech oraz wskazać na rysunku punkty obserwacji oraz wyznaczone wielkości.

1.1 Wyniki



Rysunek 1: Wykres zawierający wartości własne oraz wyznaczone wektory wartości własnych

Na wykresie zostały przedstawione punkty obserwacji i wektory wartości własnych wyznaczone za pomocą zaimplementowanej metody PCA.

1.2 Implementacja

```
1  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];
2
3  J = 2;
4
5  covX = cov(X');
6  [V, D] = eigs(covX,J);
7  Z = V'*X;
8
9  clf;
10 plot(X(1, :), (X(2, :)), '*');
11 hold on;
12 plot(Z(1, :), (Z(2, :)), '*');
13 hold on;
14 plot([0,1],V(1,:))
15 hold on;
16 plot([-1,0],[V(2,1), V(1,1)])
```

Rysunek 2: Implementacja algorytmu PCA i rysowanie wykresu dla zadanych danych

2 Zadanie 2

Drugie zadanie polegało na wyznaczeniu cech holistycznie dla różnej liczby estymowanych komponentów głównych ($J = 4, 10, 20, 30$). Obrazy oryginalne i zredukowane zostały pogrupowane metodą k-średnich. Badania zostały przeprowadzone dla liczby grup od 2 do 5 w środowisku Matlab oraz zmierzono czas i porównano dokładność grupowania. Następnie dokonano klasyfikacji obrazów w przestrzeni oryginalnej i zredukowanej przy pomocy klasyfikatora K-NN. Na koniec efekty klasyfikacji zostały porównane z efektami grupowania.



Rysunek 3: Zdjęcia twarzy, z których wybrano kilka grup do klasyfikacji

2.1 Implementacja główna

Na poniższym listingu została przedstawiona główna pętla programu. Podczas jej działania zostają wykonane następujące operacje:

- Wczytanie wybranej ilości folderów ze zdjęciami,
- Utworzenie wzorcowego wektora, zawierającego klasy kolejnych obrazów (klasa - numer osoby, której zrobiono zdjęcie),
- Dla każdej z wskazanych wartości estymowanych komponentów głównych (J) zostaje przeprowadzone zredukowanie wymiarów za pomocą funkcji *myPCA*,
- Zdjęcia oryginalne oraz zredukowane zostają poddane grupowaniu i klasyfikacji,
- Zapis skuteczności i czasu działania algorytmów do plików.

Pętla ta jest wykonywana dla różnej liczby klas (grup) zdjęć, w zależności od zmiennej *classes_count*.

```
classes_count = 5;
% Wczytywanie zdjec dla wybranej liczby klas
images_arrays = getAllImages(classes_count);
```

```

J = [4, 10, 20, 30];
iterations = 10;

% Wektor zawierajacy klasy kolejnych obrazow
images_classes = [];
for i = 1:classes_count
    images_classes = [images_classes; ones(10,1).*i];
end

% Petla glowna
for j_val = 1:4
    grouping_results = zeros([iterations 4]);
    classification_results = zeros([iterations 4]);

    for i = 1 : iterations
        % Redukcja wymiarow
        [V, pca_images_arrays, D] = myPCA(images_arrays', J(j_val));

        % Grupowanie za pomoca k-srednich
        g_result = getGroupingResults(
            images_arrays, pca_images_arrays, classes_count,
            images_classes);
        grouping_results(i,:) = g_result;

        % Klasyfikacja z uzyciem k-NN
        c_result = getClassificationResults(
            images_arrays, pca_images_arrays, classes_count,
            images_classes);
        classification_results(i,:) = c_result;
    end

    % Zapis wynikow do plikow
    (...)
end

```

W celu redukcji wymiarów za pomocą algorytmu PCA, została zaimplementowana funkcja *myPCA*, którą przedstawiono poniżej.

```

function [V, newX, D] = myPCA(X, J)
    X = bsxfun(@minus, X, mean(X,2));
    C = (X*X') ./ (size(X,2)-1);

    [V, D] = eigs(C,J);
    [D, order] = sort(diag(D), 'descend');
    V = V(:, order);
    newX = V'*X;

```

```
end
```

2.2 Implementacja grupowania

Pierwszym krokiem do rozwiązania zadania było pogrupowanie zdjęć z użyciem algorytmu k-średnich. W tym celu powstała funkcja *getGroupingResults*.

```
function result = getGroupingResults(images_arrays, pca_images_arrays,
    clusters, images_classes)

    % Uruchomienie grupowania k-srednich,
    % Zapis wyników i czasu trwania
    tic;
    groups = kmeans(images_arrays, clusters);
    default_time = toc;
    tic;
    groups_pca = kmeans(pca_images_arrays', clusters);
    pca_time = toc;

    % Obliczenie skuteczności grupowania i jej normalizacja
    default_acc = AccMeasure(images_classes, groups)/100.0;
    pca_acc = AccMeasure(images_classes, groups_pca)/100.0;

    result = [default_acc, pca_acc, default_time, pca_time];
end
```

Jak można zauważyć na powyższym listingu, funkcja przyjmuje 4 argumenty:

- *images_arrays* - oryginalne macierze zdjęć,
- *pca_images_arrays* - macierze zdjęć po redukcji wymiarów,
- *clusters* - ilość klastrów (grup) zdjęć,
- *images_classes* - wzorzec klas dla kolejnych zdjęć.

Podczas wykonywania funkcji mierzony jest czas grupowania, a następnie, za pomocą funkcji *AccMeasure*, także jej skuteczność. Numery klas przyporządkowanych kolejnym obrazom w wyniku grupowania k-średnich nie zawsze są tożsame z tymi, które zostały podane jako wzorcowe. Jednak *AccMeasure* radzi sobie z przyporządkowaniem klas założonych klasom otrzymanym ze skutecznością ponad 80%. Można to sprawdzić, uruchamiając ją w ten sposób, by zwracała trzy wartości - jedna z nich przedstawia właśnie tą skuteczność. Wyniki otrzymane za pomocą tej funkcji należy znormalizować tak, by największa możliwa ich wartość była równa 1. Wtedy można łatwo porównać je z wynikami klasyfikacji. Ostatecznie otrzymujemy skuteczność i czas grupowań zdjęć oryginalnych i tych po redukcji wymiarów.

2.3 Implementacja klasyfikacji

Kolejnym elementem implementacji jest klasyfikacja obrazów za pomocą algorytmu k najbliższych sąsiadów. W tym celu została stworzona funkcja *getClassificationResults*. Przyjmuje ona te same argumenty, co funkcja używana przy grupowaniu, jednak dla lepszego zrozumienia działania kodu zmiennej *clusters* z poprzedniego listingu odpowiada *classes_count* z obecnego.

```
function result = getClassificationResults(images_arrays, pca_images_arrays,
    classes_count, images_classes)

    % Uruchomienie klasyfikacji k-NN,
    % Zapis wyników i czasu trwania
    tic;
    model = fitcknn(images_arrays, images_classes);
    cv_model = crossval(model, 'Kfold', classes_count);
    default_time = toc;
    tic;
    model_pca = fitcknn(pca_images_arrays, images_classes);
    cv_model_pca = crossval(model_pca, 'Kfold', classes_count);
    pca_time = toc;

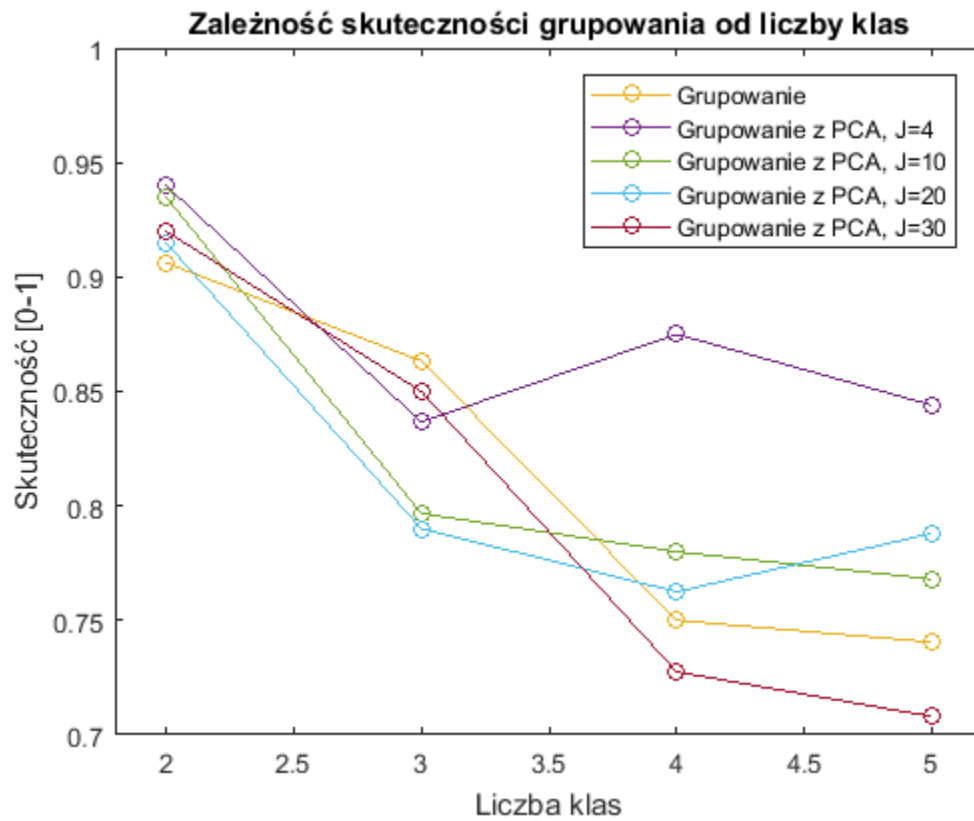
    % Obliczenie skuteczności grupowania
    cv_model_loss = kfoldLoss(cv_model);
    default_acc = 1 - cv_model_loss;
    cv_model_pca_loss = kfoldLoss(cv_model_pca);
    pca_acc = 1 - cv_model_pca_loss;

    result = [default_acc, pca_acc, default_time, pca_time];
```

Najpierw, na podstawie macierzy obrazów i wzorca ich klas, za pomocą algorytmu k-NN, tworzony jest model klasyfikatora. Następnie, z wykorzystaniem funkcji *crossval*, przeprowadzona zostaje krosvalidacja, czyli analiza modelu, polegająca na wybraniu przez algorytm różnych elementów zbioru jako zestaw uczący, a jednego jako testowy. Taki sprawdzian zostaje przeprowadzony dla wielu kombinacji elementów w grupach. Ostatnim krokiem jest sprawdzenie błędu modelu za pomocą funkcji *kfoldLoss*. Odejmując wynik od liczby 1 otrzymujemy skuteczność klasyfikacji w modelu.

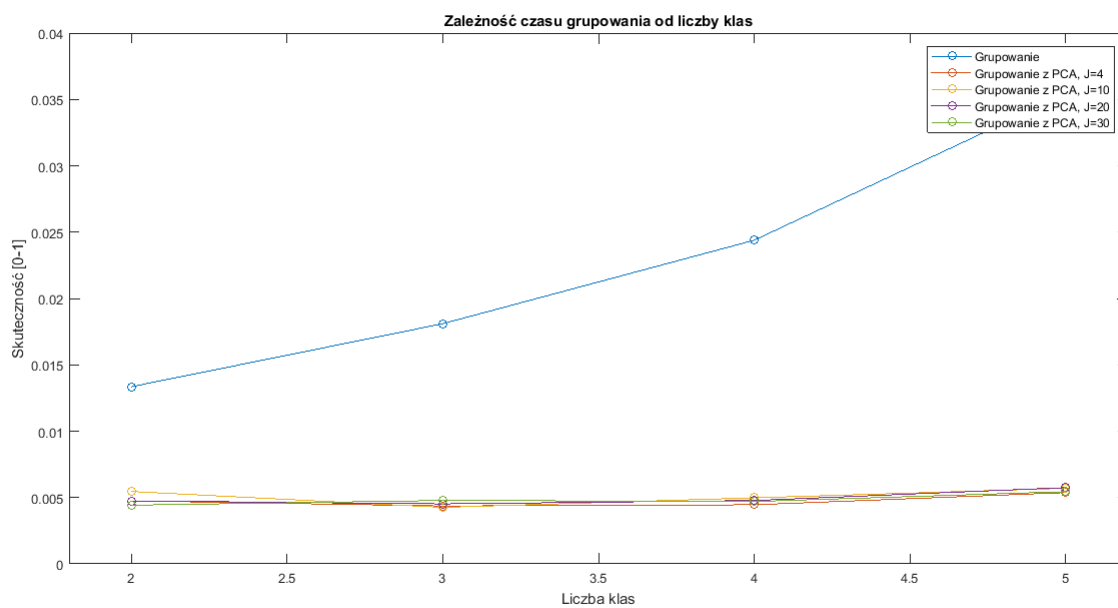
2.4 Wyniki

2.4.1 Grupowanie



Rysunek 4: Skuteczność grupowania dla wymiarów pełnych i zredukowanych

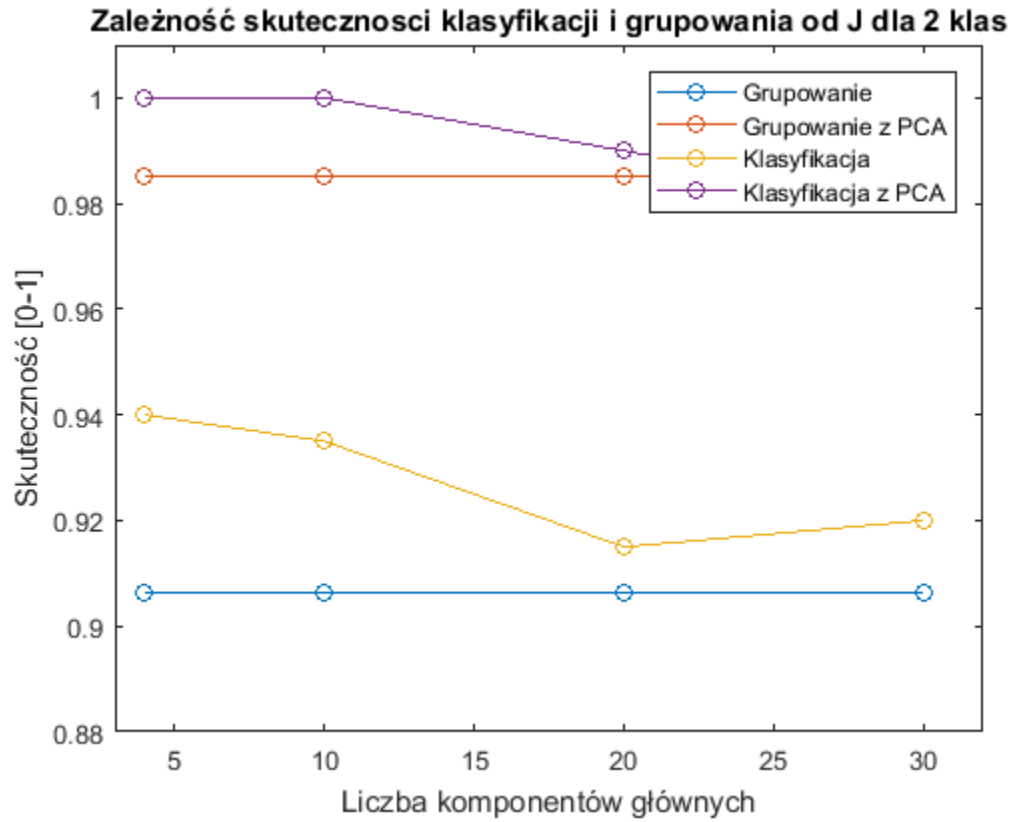
Na wykresie można zauważyć, że zwykle skuteczność grupowania spada wraz ze wzrostem ilości klas. Najgorzej w tym podsumowaniu wypadają zdjęcia po zdredukowaniu wymiarów do $J=30$, natomiast w pozytywny sposób wyróżnia się PCA z $J=4$ wymiarami.



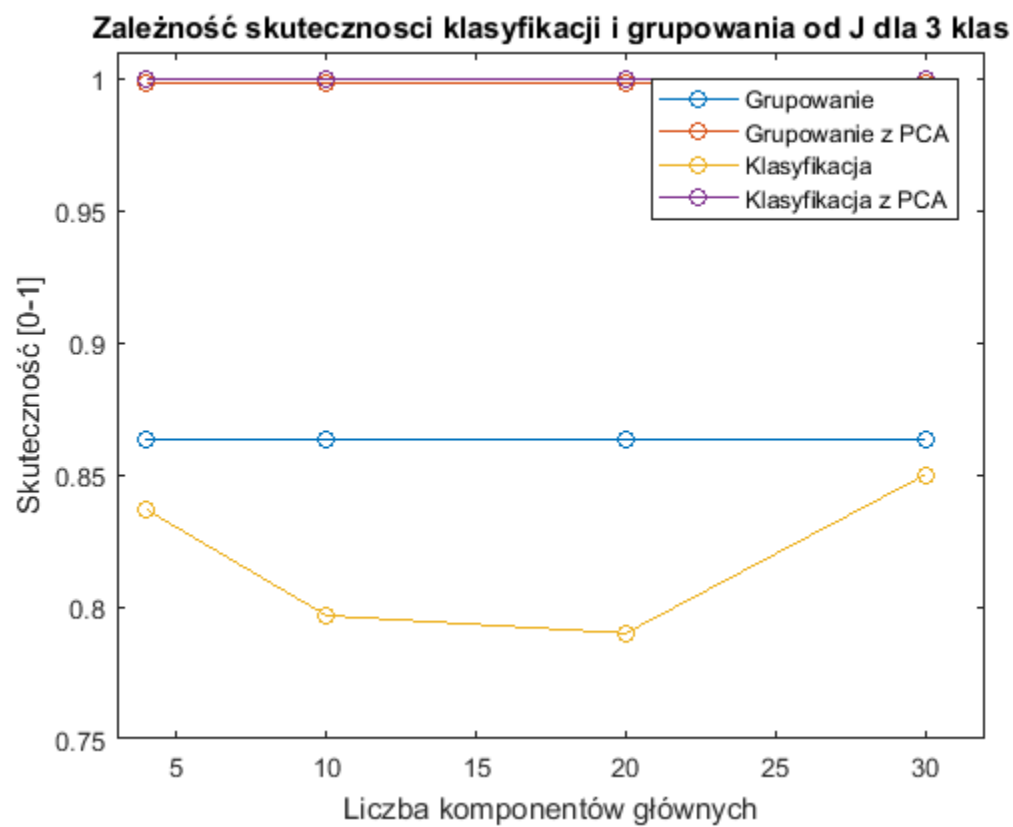
Rysunek 5: Czas grupowania dla wymiarów pełnych i zredukowanych

Na wykresie jasno widać, że redukcja wymiarów w każdym przypadku znacząco wpłynęła na redukcję czasu grupowania.

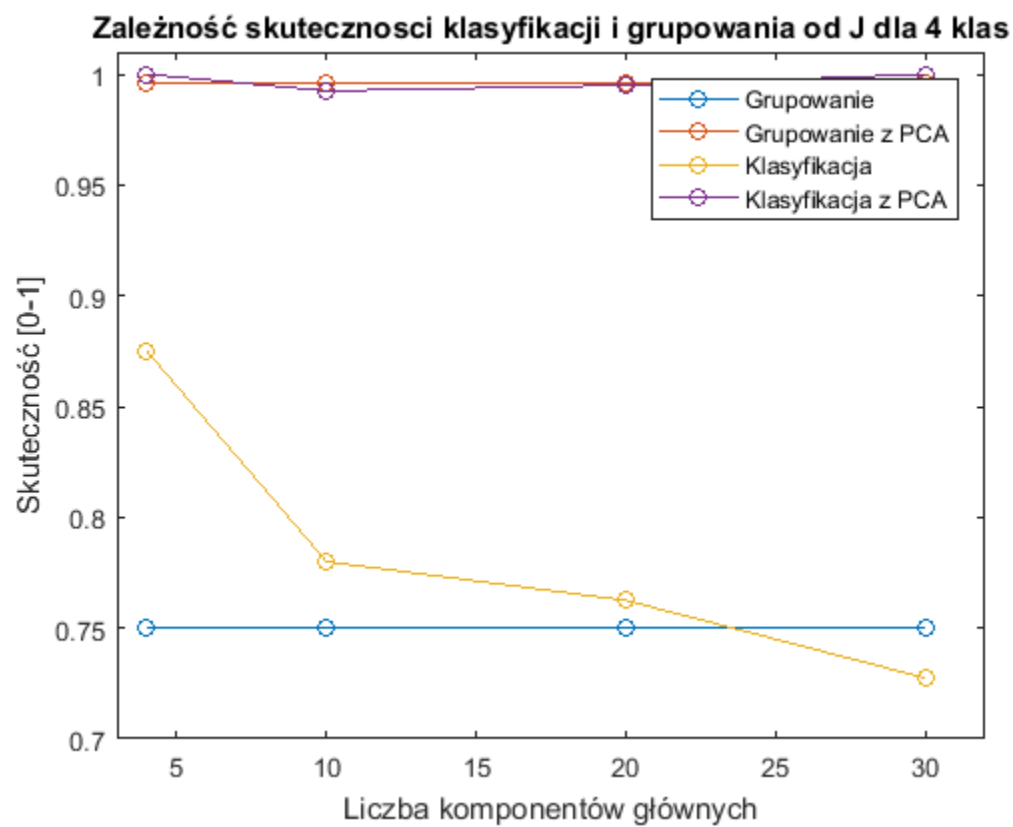
2.4.2 Porównanie klasyfikacji i grupowania



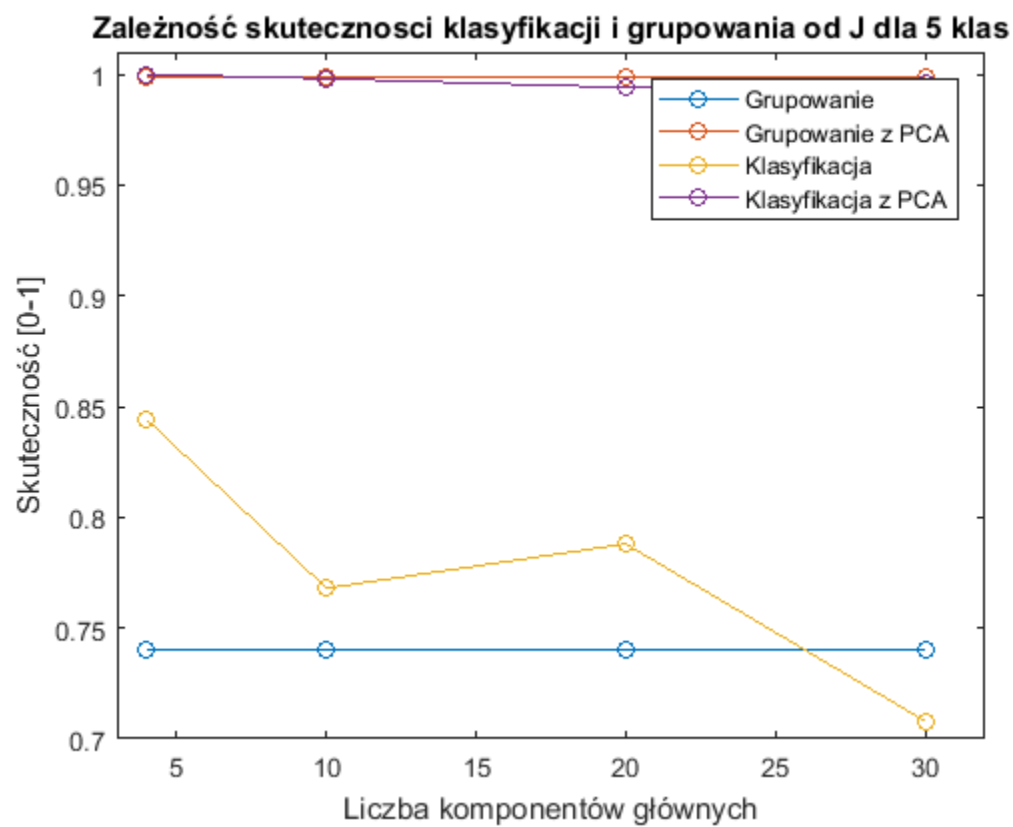
Rysunek 6: Skuteczność grupowania i klasyfikacji dla 2 klas



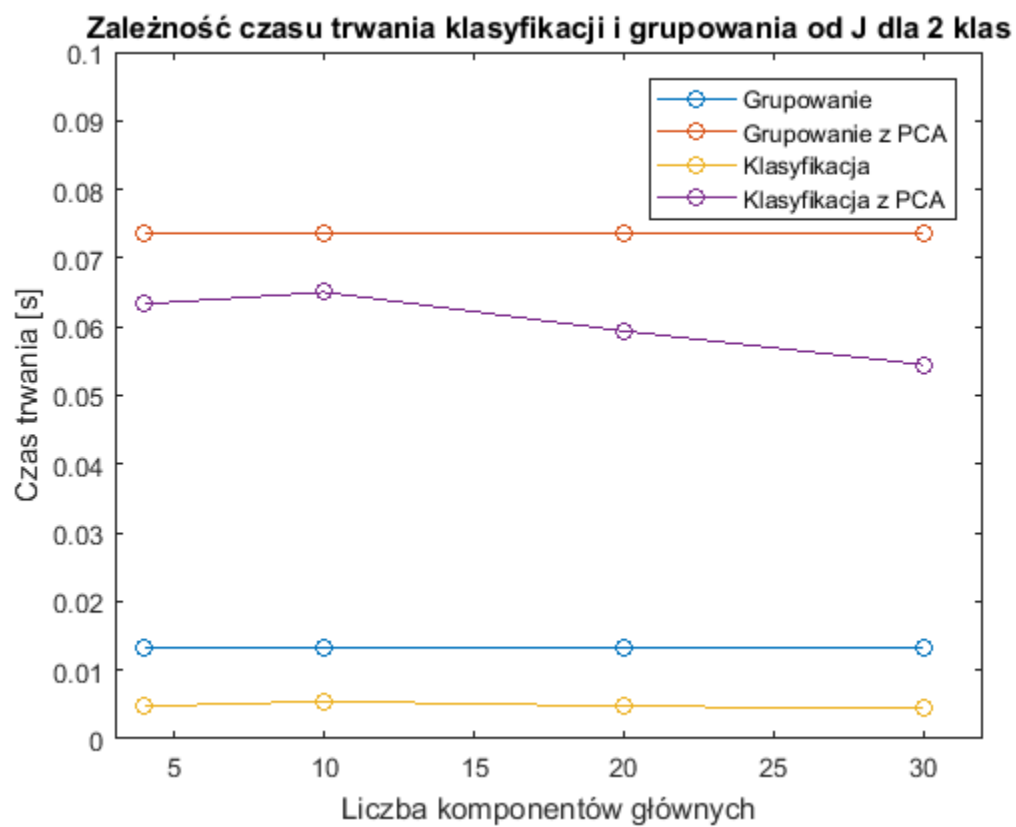
Rysunek 7: Skuteczność grupowania i klasyfikacji dla 3 klas



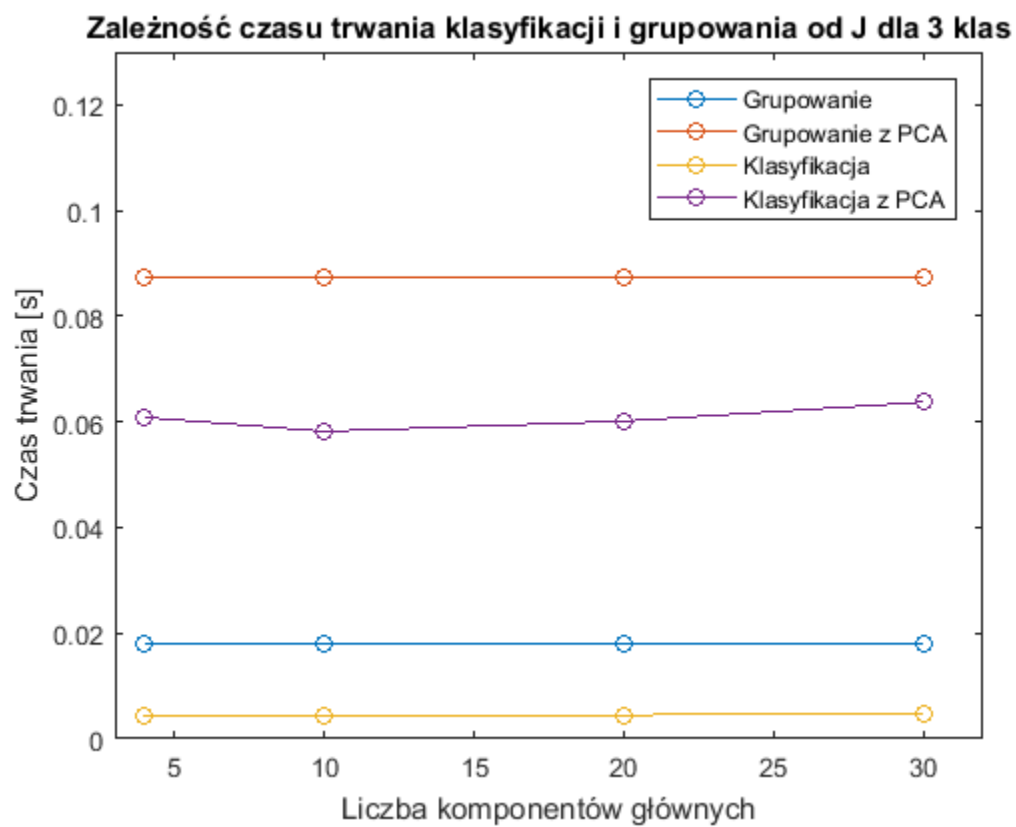
Rysunek 8: Skuteczność grupowania i klasyfikacji dla 4 klas



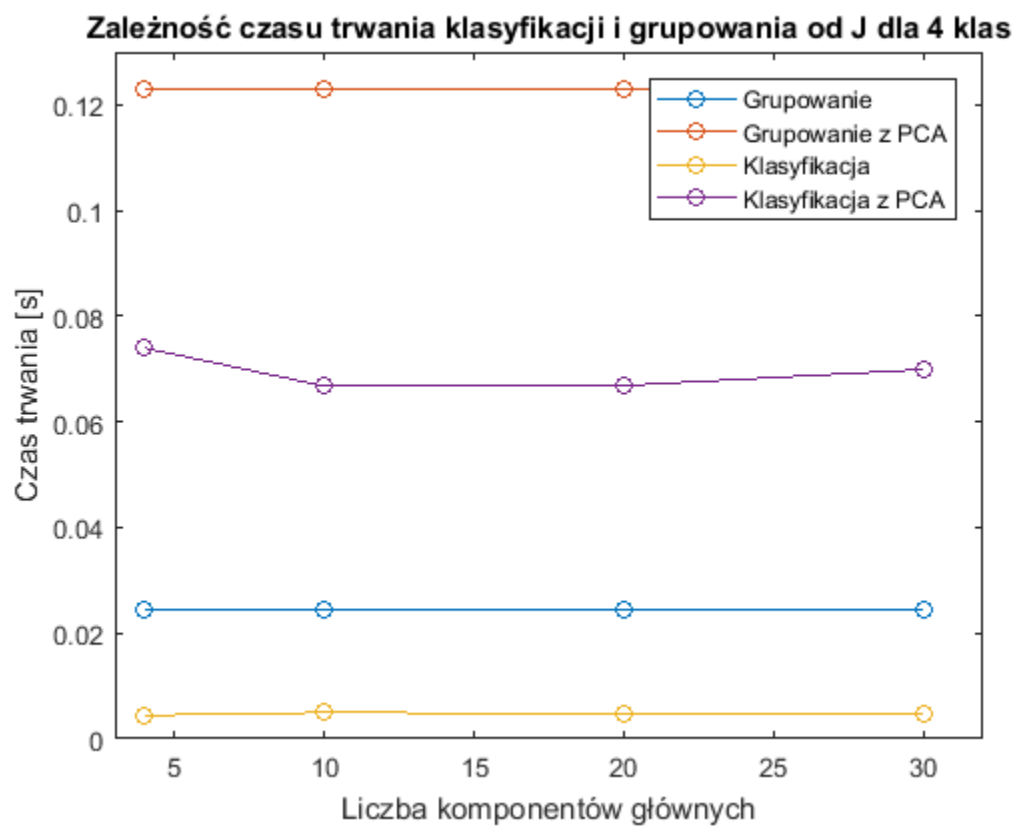
Rysunek 9: Skuteczność grupowania i klasyfikacji dla 5 klas



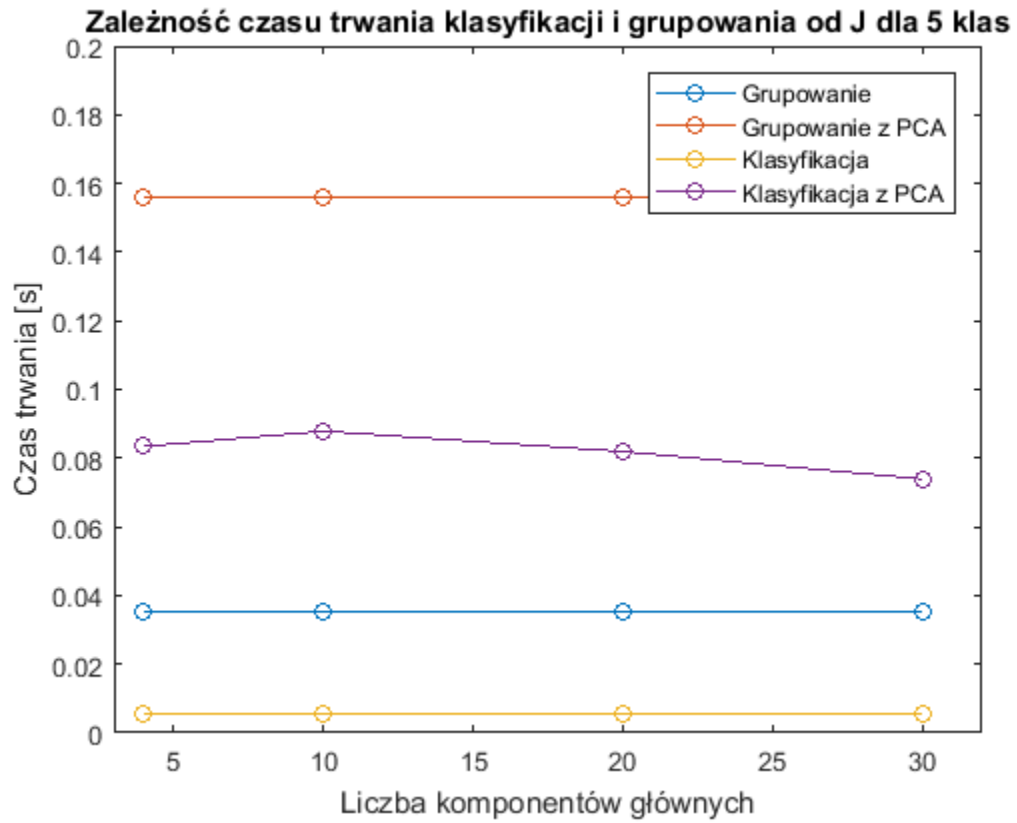
Rysunek 10: Czas grupowania i klasyfikacji dla 2 klas



Rysunek 11: Czas grupowania i klasyfikacji dla 3 klas



Rysunek 12: Czas grupowania i klasyfikacji dla 4 klas



Rysunek 13: Czas grupowania i klasyfikacji dla 5 klas

Zarówno w przypadku grupowania, jak i klasyfikacji, zastosowanie PCA korzystnie wpływa na skuteczność działania algorytmów. Jeśli chodzi o czas wykonywania programu, jest on dłuższy przy użyciu obrazów zredukowanych przez algorytm PCA.