# Matematyczne aspekty analizy danych
## semestr zimowy 2025/2026

Dr Anna Muranova
UWM w Olsztynie

Wykład 13

# Rozdział 11. Redukcja wymiaru

# Wprowadzenie do metod redukcje wymiaru

Bazy danych zwykle używane w eksploracji danych mogą zawierać miliony rekordów i tysiące zmiennych. Jest mało prawdopodobne, aby wszystkie zmienne były niezależne i nie mieli korelacje pomiędzy sobą.
Metody redukcji wymiarów mają na celu wykorzystanie struktury korelacji wśród zmiennych predykcyjnych, aby osiągnąć następujące cele:

- zmniejszyć liczbę komponentów predykcyjnych,
- zapewnić niezależność tych komponentów,
- zapewnić zasady dla interpretacji wyników.

## Metody redukcji wymiarów:

- Analiza składowych głównych (PCA, ang. Principal Components Analysis)
- Nieujemna faktoryzacja macierzy
- Jądro PCA (Kernel PCA)
- Jądro PCA oparte na grafach (Graph-based kernel PCA)
- Liniowa analiza dyskryminacyjna (LDA, Linear discriminant analysis)
- Uogólniona analiza dyskryminacyjna (GDA, Generalized discriminant analysis)
- Nieliniowe skalowanie wielowymiarowe (Sammon Mapping)
- Skalowanie wielowymiarowe Kruskalla (MDS, ang. Multidimensional Scaling)

## Analiza składowych głównych

Analiza głównych składowych ma na celu wyjaśnienie struktury korelacji zestawu zmiennych predykcyjnych przy użyciu mniejszego zestawu kombinacji liniowych tych zmiennych. Te kombinacje liniowe nazywane są komponentami. Całkowita zmienność zbioru danych tworzona przez pełny zestaw $m$ zmiennych często można wyjaśnić przy pomocy mniejszego zbioru $k$ kombinacji liniowych tych zmiennych. Tzn., w składowych $k$ znajduje się prawie tyle samo informacji, jak w oryginalnych zmiennych. Zatem w razie potrzeby analityk może następnie zastąpić oryginalne $m$ zmiennych $k < m$, co jest przydatne dla dużych zbiorów danych. W takim razie nowy zbiór danych skalda się z $k$ liczbowych danych dla każdej z $n$ obserwacje, zamiast $m$ danych dla każdej z $n$ obserwacje.
Jako dane wejściowe do algorytmu PCA podawana jest macierz zawierająca kolejne obserwacje, na podstawie których będą wyznaczane główne składowe (wektory bazowe nowej przestrzeni). Są one zwracane również jako jedna macierz.

## Algorytm PCA

Niech początkowe dane zapisane są w postaci macierze:

$$X = \begin{pmatrix} \mathrm{X}_1^{\mathrm{T}} \\ \mathrm{X}_2^{\mathrm{T}} \\ \vdots \\ \mathrm{X}_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1m} \\ X_{21} & X_{22} & \dots & X_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{nm} \end{pmatrix},$$

Algorytm PCA składa się z następujących kroków:

1. Wyznaczenie średnich i wariancje dla kolumn

Jest to pierwsza czynność konieczna do stworzenia macierzy kowariancji macierzy wejściowej. Matematycznie można ten krok zapisać jako:

$$u_j = \frac{1}{n}\sum_{i=1}^{n} X_{ij},$$

$$w_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_{ij} - u_j)^2}$$

Kolejne pozycje wektora średnich $u$ przechowują więc średnie odpowiadających kolumny. Obliczane są więc średnie wartości kolejnych cech dla wszystkich obserwacji.

# Algorytm PCA

2. Wyliczanie macierzy odchyleń
Krok ten polega na odjęciu od macierzy wejściowej średnich wyliczonych w punkcie 1. Od każdego elementu macierzy odejmujemy średnią dla kolumny, w którym się znajduje:

$$B_{ij} = \frac{X_{ij} - u_j}{w_j}.$$

3. Wyznaczenie macierzy kowariancji
W ogólnym przypadku macierz kowariancji wylicza się ze wzoru:

$$\mathrm{C} = \frac{1}{n} B^* B,$$

gdzie $\mathrm{B} = (B_{ij})$ to macierz odchyleń. W przypadku, gdy wartości macierzy B są rzeczywiste, użyte we wzorze sprzężenie hermitowskie $(^*)$ jest tożsame ze zwykłą transpozycją.

## Algorytm PCA

4. Obliczenie wartości własnych macierzy kowariancji
Wyliczenie macierzy V wektorów własnych znormalizowanych ($V^{-1} = V^T$), która spełnia:

$$V^{-1}CV = D,$$

gdzie D jest macierzą diagonalną wartości własnych C. Ten krok zazwyczaj wymaga użycia algorytmu komputerowego do obliczania wektorów własnych i wartości własnych.

- Macierz D będzie miała postać macierzy diagonalnej $m \times m$, gdzie

  $$D_{ij} = \lambda_i \qquad \text{for } j = i$$

  jest $i$-tą wartością własną macierzy kowariancji **C**, oraz

  $$D_{ij} = 0 \qquad \text{for } i \neq j.$$

- Macierz V, również o wymiarze $m \times m$, zawiera $m$ wektorów kolumnowych, każdy o długości $m$ które reprezentują wektory własne (prawe) macierzy kowariancji C. Wartości własne i wektory własne są uporządkowane i sparowane. $j$-ta wartość własna odpowiada $j$-temu wektorowi własnemu.

5. Zmiana układu wektorów własnych i wartości własnych
Na tym etapie można dokonać zawężenia wymiaru przestrzeni. Z otrzymanych wartości własnych wybieramy te największe, co ma na celu minimalizację straty informacji podczas rzutowania danych na mniejszą liczbę wymiarów. Im wyższa wartość własna tym odpowiadający jej wektor własny jest słabiej skorelowany z pozostałymi.
Kolumny macierzy wektorów własnych V i macierzy wartości własnych D trzeba posortować w kolejności malejącej wartości własnej. Trzeba pamiętać, żeby zachować prawidłowe sparowanie pomiędzy kolumnami w każdej macierzy.

## Algorytm PCA

6. Obliczenie skumulowanej zawartości energii dla każdego wektora własnego. Wartości własne reprezentują rozkład energii danych źródłowych wśród każdego z wektorów własnych, przy czym wektory własne stanowią bazą przestrzeni liniowej danych. Skumulowana zawartość energii $g$ dla $j$-tego wektora własnego jest sumą zawartości energii we wszystkich wartościach własnych od 1 do $j$:

$$g_j = \sum_{k=1}^{j} D_{kk} \qquad \text{for } j = 1, \ldots, m$$

## Algorytm PCA

7. Wybranie podzbioru wektorów własnych jako wektorów bazowych. Trzeba wybrać $L, L < m$ pierwszych kolumn macierzy V jako macierz W o pomiarach $m \times L$:

$$W_{ij} = V_{ij} \qquad \text{for } i = 1, \ldots, m \qquad j = 1, \ldots, L$$

Wektora $g$ używa się jako wskazówka przy wyborze odpowiedniej wartości $L$. Celem jest wybranie jak najmniejszej wartości $L$ przy jednoczesnym osiągnięciu stosunkowo dużej wartości $g$ w ujęciu procentowym, napr.

$$\frac{g_L}{g_m} \geq 0.9.$$

## Algorytm PCA

8. Rzut danych na nową bazą.

Rzutowane punkty danych to wiersze macierzy

$$T = BW,$$

Oznacza to, że pierwsza kolumna T to rzut punktów danych na pierwszą składową główną, druga kolumna to rzut na drugą składową główną itd.

## Przykład

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 5 & 6 & 7 \\ 1 & 4 & 2 & 3 \\ 5 & 3 & 2 & 1 \\ 8 & 1 & 2 & 2 \end{pmatrix}$$

## Przykład: Python

```
import numpy as np

X = np.array([[1,2,3,4],[5,5,6,7],[1,4,2,3],[5,3,2,1],[8,1,2,2]])
m = len(X.transpose())
n = len(X)
u =np.mean(X, axis=0)#0 - by column
w = np.array([(np.var(X[:,i]))**(1/2) for i in range(m)])
U = np.meshgrid(u,np.ones(n))[0]
W = np.meshgrid(w,np.ones(n))[0]
B = (X - U)/W
print(B)
C = 0.2*B.transpose()@B
V = np.linalg.eig(C)[1]#eigenvectors
print(V)
print(np.linalg.eig(C)[0])#eigenvalues
D = np.linalg.inv(V)@C@V
print(D.round(2))
W = V[:,0:2]
T = B@W
print(T.round(2))
```

## Przykład: Python

```
import numpy as np

X = np.array([[1,2,3,4],[5,5,6,7],[1,4,2,3],[5,3,2,1],[8,1,2,2]])
m = len(X.transpose())
n = len(X)
u =np.mean(X, axis=0)#0 - by column
w = np.array([(np.var(X[:,i]))**(1/2) for i in range(m)])
U = np.meshgrid(u,np.ones(n))[0]
W = np.meshgrid(w,np.ones(n))[0]
B = (X - U)/W
print(B)
C = 0.2*B.transpose()@B
V = np.linalg.eig(C)[1]#eigenvectors
print(V)
print(np.linalg.eig(C)[0])#eigenvalues
D = np.linalg.inv(V)@C@V
print(D.round(2))
W = V[:,0:2]
T = B@W
print(T.round(2))
```

## Przykład: Python pingwiny

```
import numpy as np
X =  np.array([ [39.5, 40.3, 36.7, 38.9, 41.1],
                    [17.4, 18, 19.3, 17.8, 17.6],
                    [186, 195, 193, 181, 182],
                    [3.8, 3.25, 3.45, 3.625, 3.2]]).transpose()
m = len(X.transpose())
n = len(X)
u =np.mean(X, axis=0)#0 - by column
w = np.array([(np.var(X[:,i]))**(1/2) for i in range(m)])
U = np.meshgrid(u,np.ones(n))[0]
W = np.meshgrid(w,np.ones(n))[0]
B = (X - U)/W
C = 0.2*B.transpose()@B
V = np.linalg.eig(C)[1]#eigenvectors
print(V)
print(np.linalg.eig(C)[0])#eigenvalues
D = np.linalg.inv(V)@C@V
print(D.round(2))
W = V[:,0:2]
T = B@W
print(T.round(2))
```

## Przykład: Python, sklearn

```
import numpy as np
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA

X = np.array([[1,2,3,4],[5,5,6,7],[1,4,2,3],[5,3,2,1],[8,1,2,2]])
#X =  np.array([ [39.5, 40.3, 36.7, 38.9, 41.1],[17.4, 18, 19.3, 17.8,
#[186, 195, 193, 181, 182],[3.8, 3.25, 3.45, 3.625, 3.2]]).transpose()

pca = PCA(n_components=2)
pca.fit(scale(X))
print(pca.components_)#e.v
print(sum(pca.explained_variance_ratio_))#%of information
print(pca.transform(scale(X)).round(2))
```

## Przykład: Python, sklearn, dataframe pingwiny

```
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
import pandas as pd

data = pd.DataFrame({
    'Dl_dz': [39.5, 40.3, 36.7, 38.9, 41.1],
    'Sz_dz': [17.4, 18, 19.3, 17.8, 17.6],
    'Dl_pl': [186, 195, 193, 181, 182],
    'waga': [3.8, 3.25, 3.45, 3.625, 3.2]
})

newdata = pd.DataFrame(scale(data), index=data.index,
                columns=data.columns)
print(newdata)
pca = PCA(n_components=2)
pca.fit(newdata)
print(pca.components_)#e.v
print(sum(pca.explained_variance_ratio_))#%of information
print(pca.transform(newdata).round(2))
```

## Przykład: wszystkie pingwiny

```
import seaborn as sns
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
import pandas as pd
import matplotlib.pyplot as plt
data = sns.load_dataset("penguins").dropna()
numdata = data[["bill_length_mm","bill_depth_mm",
                 "flipper_length_mm", "body_mass_g"]]
newdata = pd.DataFrame(scale(numdata), index=numdata.index,
                columns=numdata.columns)
pca = PCA(n_components=2)
pca.fit(newdata)
print(sum(pca.explained_variance_ratio_))#%of information
transf =  pca.transform(newdata).round(2)
print(transf)
df = data[["species","island", "sex"]]
df['First component']=transf[:,0]
df['Second component']=transf[:,1]
sns.scatterplot(data=df, x="First component",
      y="Second component", hue="sex", style = "species")
plt.show()
```

## Przykład: wszystkie pingwiny