Metoda PCA na podstawie SVD

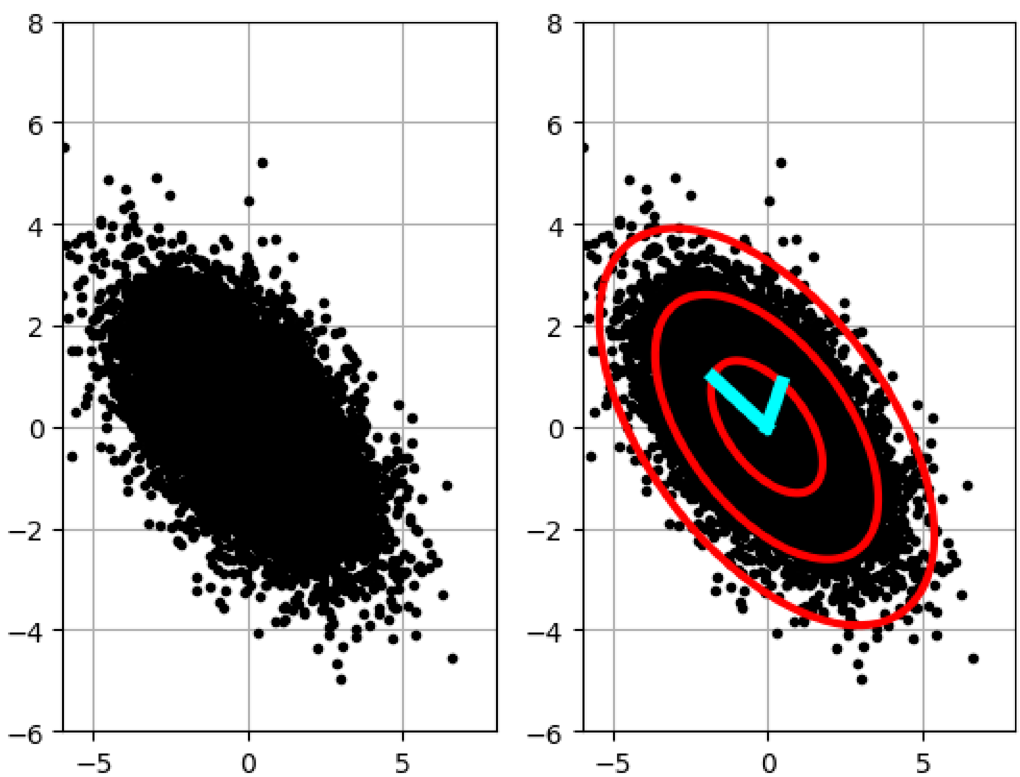
December 29, 2023

Sprawozdanie
Matematyka Konkretna
Prowadzący: prof. dr hab. Vasyl Martsenyuk
Laboratorium 4
30.10.2023
Metoda PCA na podstawie SVD
Wariant 10
Link do repozytorium: https://github.com/Maksiolo20/MK

[5]:

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
# Read data from the CSV file
data = pd.read_csv('10.csv', sep=',')
# Extract the first row as the center of the data and principal axes
center_and_axes = data.values.flatten()
# Extracting values for the center of the data and principal axes
center = center_and_axes[:1000] # Assuming the first 1000 values are for the␣
 ↪center
axes = center_and_axes[1000:]
theta = np.pi / 3 # Rotate cloud by pi/3
R = np.array([[np.cos(theta), -np.sin(theta)], # Rotation matrix
[np.sin(theta), np.cos(theta)]])
nPoints = 10000 # Create 10,000 points
sig = np.array([1.0, 2.0])  # Replace these values with your desired standard␣
 ↪deviations
xC = np.array([0.0, 0.0])  # Replace these values with your desired center
X = R @ np.diag(sig) @ np.random.randn(2, nPoints) + np.diag(xC) @ np.ones((2,␣
 ↪nPoints))
fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.plot(X[0, :], X[1, :], '.', color='k')
ax1.grid()
plt.xlim((-6, 8))
plt.ylim((-6, 8))
Xavg = np.mean(X, axis=1) # Compute mean
B = X - np.tile(Xavg, (nPoints, 1)).T # Mean-subtracted data
# Find principal components (SVD)
```

```python
U, S, VT = np.linalg.svd(B / np.sqrt(nPoints), full_matrices=0)
ax2 = fig.add_subplot(122)
ax2.plot(X[0, :], X[1, :], '.', color='k') # Plot data to overlay PCA
ax2.grid()
plt.xlim((-6, 8))
plt.ylim((-6, 8))
theta = 2 * np.pi * np.arange(0, 1, 0.01)
# 1-std confidence interval
Xstd = U @ np.diag(S) @ np.array([np.cos(theta), np.sin(theta)])
ax2.plot(Xavg[0] + Xstd[0, :], Xavg[1] + Xstd[1, :], '-', color='r',
linewidth=3)
ax2.plot(Xavg[0] + 2 * Xstd[0, :], Xavg[1] + 2 * Xstd[1, :], '-',
color='r', linewidth=3)
ax2.plot(Xavg[0] + 3 * Xstd[0, :], Xavg[1] + 3 * Xstd[1, :], '-',
color='r', linewidth=3)
# Plot principal components U[:,0]S[0] and U[:,1]S[1]
ax2.plot(np.array([Xavg[0], Xavg[0] + U[0, 0] * S[0]]),
np.array([Xavg[1], Xavg[1] + U[1, 0] * S[0]]), '-',
color='cyan', linewidth=5)
ax2.plot(np.array([Xavg[0], Xavg[0] + U[0, 1] * S[1]]),
np.array([Xavg[1], Xavg[1] + U[1, 1] * S[1]]), '-',
color='cyan', linewidth=5)
plt.show()
```



[ ]:

```
```