SPRAWOZDANIE

Zajęcia: Matematyka konkretna

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 1 Data 11.10.2025 **Temat:** „Analiza macierzowa. Podstawowe poje¸cia. Rozk lad SVD” Wariant 1	Marika Daniszewska Informatyka II stopień, stacjonarne, 2 semestr, gr. 1a

1. Polecenie: wariant 1 zadania

W ramach laboratorium należało wykonać zadanie dotyczące kompresji obrazu metodą SVD (Singular Value Decomposition).
Celem było zbadanie, w jaki sposób rozkład SVD może być wykorzystany do redukcji danych obrazowych przy zachowaniu możliwie wysokiej jakości obrazu.

W szczególności należało:

- wczytać dowolny obraz i przekształcić go do skali szarości,
- przeprowadzić rozkład SVD dla macierzy obrazu,
- obliczyć, ile wartości singularnych należy zachować, aby odtworzony obraz zachował 90% energii (informacji),
- porównać wyniki rekonstrukcji obrazu dla różnych wartości `k` (liczby zachowanych wartości singularnych),
- obliczyć przybliżony współczynnik kompresji oraz przeanalizować wpływ kompresji na jakość obrazu.

2. Opis programu opracowanego (kody źródłowe, zrzuty ekranu)

- Wczytanie obrazu

Obraz (`1.webp`) jest wczytywany z pliku i konwertowany do skali szarości.
Macierz obrazu jest zapisywana jako tablica typu `numpy.ndarray`.

- Rozkład SVD

Program wykonuje rozkład:

A=UΣVTA = U \Sigma V^TA=UΣVT

gdzie `A` to macierz obrazu, `U` i `V` – macierze ortogonalne, a `Σ` – macierz diagonalna zawierająca wartości singularne.

- Analiza energii

Obliczana jest suma kwadratów wartości singularnych oraz wykres skumulowanej energii. Na jego podstawie określane jest minimalne `k`, które zapewnia zachowanie 90% całkowitej energii obrazu.

- Rekonstrukcja i porównanie

Program rekonstruuje obraz przy różnych wartościach `k` (np. 5, 20, 50, 100, k_{90}, 200). Każdy zrekonstruowany obraz jest zapisywany do katalogu `svd_results/` i prezentowany na jednym wykresie obok oryginału.

- Obliczenie współczynnika kompresji

Szacowany współczynnik kompresji określa stosunek liczby pikseli w oryginalnym obrazie do liczby parametrów potrzebnych do zapisu obrazu w formie zredukowanego rozkładu SVD.

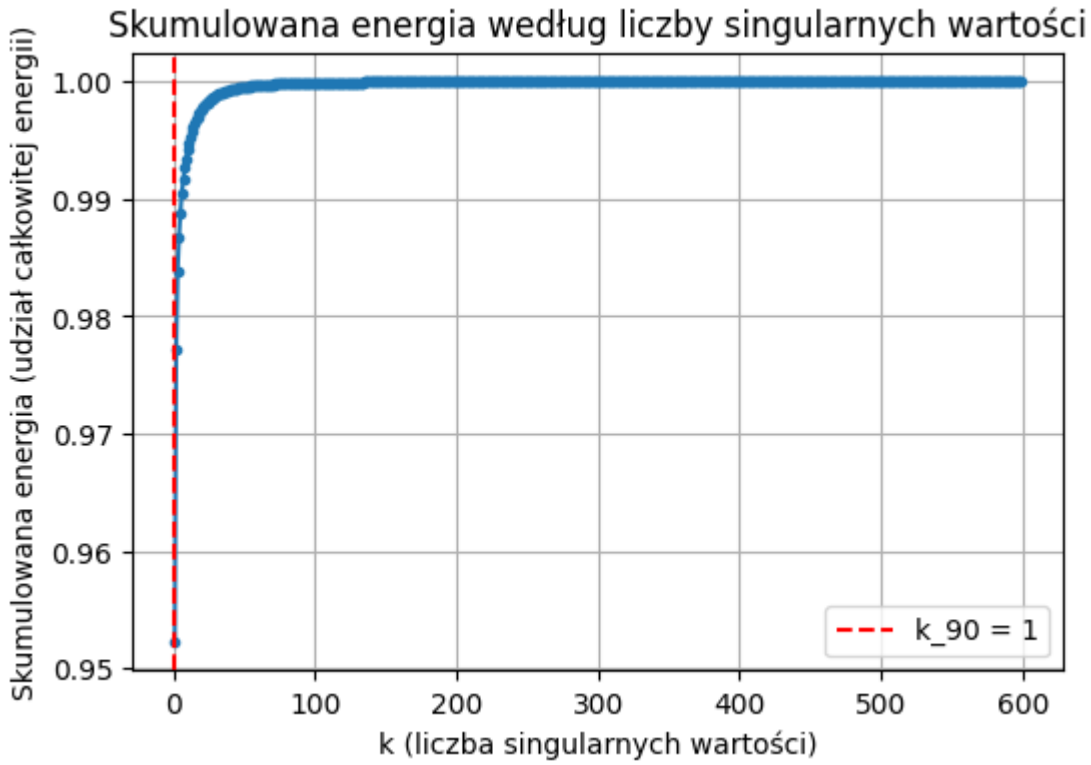
```python
# Importy i funkcje pomocnicze
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os
from pathlib import Path

# Funkcja do wczytania obrazu i konwersji do skali szarości (float)
def load_image_gray(path):
    img = Image.open(path).convert('L')  # 'L' -> grayscale
    arr = np.array(img).astype(float)
    return arr, img.size  # size = (width, height)

# Funkcja do zapisu macierzy jako obrazu uint8
def save_image_from_array(arr, out_path):
    arr_clipped = np.clip(arr, 0, 255).astype(np.uint8)
    Image.fromarray(arr_clipped).save(out_path)

# Funkcja do rekonstrukcji przy użyciu k wartości singularnych
def svd_reconstruct(A, k):
    U, S, Vt = np.linalg.svd(A, full_matrices=False)
    Uk = U[:, :k]
    Sk = np.diag(S[:k])
    Vtk = Vt[:k, :]
    return Uk @ Sk @ Vtk

# Funkcja do obliczenia minimalnego k by zachować energy_ratio (np. 0.9)
def find_k_for_energy(S, energy_ratio=0.9):
    singular_vals_sq = S**2
    total_energy = singular_vals_sq.sum()
    cumulative = np.cumsum(singular_vals_sq)
```

```
    k = np.searchsorted(cumulative, energy_ratio * total_energy) + 1
    return k, total_energy, cumulative

# Ścieżka do obrazu
image_path = '1.webp'
out_dir = 'wyniki'
os.makedirs(out_dir, exist_ok=True)

A, size = load_image_gray(image_path)
h, w = A.shape
print(f"Rozmiar obrazu (h, w): {A.shape}, rozmiar z Pillow (width, height):
{size}")

# Pokaż oryginalny (grayscale)
plt.figure(figsize=(6,6))
plt.imshow(A, cmap='gray')
plt.title('Oryginalny obraz (skala szarości)')
plt.axis('off')
plt.show()

# Obliczanie SVD (full_matrices=False dla wydajności)
U, S, Vt = np.linalg.svd(A, full_matrices=False)
print('Liczba singularnych wartości:', S.shape[0])

# Znajdź k dla 90% energii
energy_target = 0.90
k_90, total_energy, cumulative = find_k_for_energy(S,
energy_ratio=energy_target)
print(f"Minimalne k potrzebne by zachować {energy_target*100:.0f}% energii:
{k_90} z {S.shape[0]}")

# Wykres skumulowanej energii
plt.figure(figsize=(6,4))
plt.plot(np.arange(1, len(S)+1), np.cumsum(S**2) / total_energy, marker='o',
markersize=3)
plt.xlabel('k (liczba singularnych wartości)')
plt.ylabel('Skumulowana energia (udział całkowitej energii)')
plt.grid(True)
plt.axvline(k_90, color='r', linestyle='--', label=f'k_90 = {k_90}')
plt.legend()
plt.title('Skumulowana energia według liczby singularnych wartości')
plt.show()

# Wybrane wartości k do demonstracji: małe, k_90, większe (ograniczone do
długości S)
k_values = [5, 20, k_90, 50, 100, min(len(S), 200)]
k_values = sorted(list(set([min(int(k), len(S)) for k in k_values])))
recons = {}
```

```
for k in k_values:
    rec = svd_reconstruct(A, k)
    recons[k] = rec
    out_path = os.path.join(out_dir, f'recon_k_{k}.png')
    save_image_from_array(rec, out_path)
    print(f"Zapisano rekonstrukcję dla k={k} -> {out_path} (rozmiar:
{rec.shape})")

# Pokaż porównanie (ograniczone do max 6 kolumn)
cols = len(recons) + 1
plt.figure(figsize=(3*cols, 4))
plt.subplot(1, cols, 1)
plt.imshow(A, cmap='gray')
plt.title('Oryginał')
plt.axis('off')

i = 2
for k, rec in sorted(recons.items()):
    plt.subplot(1, cols, i)
    plt.imshow(rec, cmap='gray')
    plt.title(f'k={k}')
    plt.axis('off')
    i += 1
plt.tight_layout()
plt.show()

# Obliczanie współczynnika kompresji dla k (przybliżonego)
def compression_ratio(h, w, k):
    original = h * w
    compressed = h*k + k + k*w
    return original / compressed

cr_k90 = compression_ratio(h, w, k_90)
print(f'Przybliżony współczynnik kompresji dla k={k_90}: {cr_k90:.3f}
(oryginalne piksele / parametry SVD)')
# Zapisz macierz rekonstrukcji dla k_90 (już powinna być zapisana)
recon_k90_path = os.path.join(out_dir, f'recon_k_{k_90}.png')
print('Ścieżka do zapisanego pliku rekonstrukcji k_90:', recon_k90_path)

# Lista wygenerowanych plików w katalogu wyników
import glob
files = glob.glob(os.path.join(out_dir, '*'))
for f in files:
    print('-', f)
```

```
Rozmiar obrazu (h, w): (599, 800), rozmiar z Pillow (width, height): (800,
599)
```


```
Liczba singularnych wartości: 599
Minimalne k potrzebne by zachować 90% energii: 1 z 599
```




```
Przybliżony współczynnik kompresji dla k=1: 342.286 (oryginalne piksele /
parametry SVD)
```

Github: https://github.com/MarikaDan/MK/tree/main/LAB1

3. Wnioski

- Metoda SVD pozwala w efektywny sposób kompresować obrazy, zachowując dużą część informacji wizualnej przy znacznej redukcji danych.

- Jakość obrazu po rekonstrukcji zależy bezpośrednio od liczby zachowanych wartości singularnych `k`.

  - Dla małego `k` obraz jest mocno rozmazany i traci szczegóły.
  - Dla większego `k` obraz staje się coraz bardziej podobny do oryginału.

- Wartości singularne o największej wartości zawierają najwięcej informacji o strukturze obrazu, a kolejne odpowiadają za szczegóły i szum.

- Zachowanie około 90% energii (informacji) często pozwala uzyskać wizualnie bardzo dobrą jakość przy znacznie mniejszym rozmiarze danych.

- Metoda ta może być z powodzeniem stosowana w praktyce do kompresji obrazów, redukcji wymiarowości danych oraz wstępnego przetwarzania w zadaniach uczenia maszynowego.