Laboratorium 3.

Rekurencyjna kompresja macierzy

Bartosz Hanc

1 Wstęp

Celem ćwiczenia było napisanie rekurencyjnej kompresji macierzy z wykorzystaniem algorytmu częściowego SVD oraz wizualizacji uzyskiwanych w tej procedurze macierzy hierarchicznych. Dodatkowo zmierzono czas kompresji macierzy oraz błąd względny między macierzą wejściową, a macierzą uzyskaną po dekompresji dla różnych minimalnych wartości osobliwych i maksymalnego rank.

2 Kod rozwiązania

Poniżej zamieszczono kod rozwiązania w języku Python. Węzły drzewa były reprezentowane jako struktury `Node`, których definicję zawarto poniżej. Funkcja `CreateTree()` odpowiada za stworzenie drzewa kompresji, a `CompressMatrix()` za skompresowanie liścia korzystając z algorytmu SVD.

```
class Node:
    def __init__(self, rank=None, side=None, sMin=None,
                 tMin=None, U=None, V=None, D=None):
        self.next: list[Node] = []
        self.sMin: int = sMin
        self.tMin: int = tMin
        self.side: int = side
        self.rank: int = rank
        self.U: np.ndarray[float] = U
        self.D: np.ndarray[float] = D
        self.V: np.ndarray[float] = V
```

```
def CompressMatrix(A, U, D, V: np.ndarray[float], r, sMin, tMin):
    if np.allclose(A, np.zeros(A.shape)):
        return Node(rank=0, side=A.shape[0], sMin=sMin, tMin=tMin)

    return Node( rank=r, side=A.shape[0], sMin=sMin, tMin=tMin,
            U=U[:, :r], D=D[:r], V=V[:r, :])
```

```
def CreateTree(A, r, eps, sMin = 0, tMin = 0):
    n = A.shape[0]
    U, D, V = randomized_svd(A, r + 1) # from sklearn.utils.extmath
    if len(D) <= r or D[r] < eps:
        v = CompressMatrix(A, U, D, V, len(D), sMin, tMin)
    else:
        v = Node(rank=None, side=n, sMin=sMin, tMin=tMin)
        v.next.append(CreateTree(A[: n // 2, : n // 2],
                        r, eps, sMin, tMin))

        v.next.append(CreateTree(A[n // 2 :, : n // 2],
                        r, eps, sMin + n // 2, tMin))

        v.next.append(CreateTree(A[: n // 2, n // 2 :],
                        r, eps, sMin, tMin + n // 2))

        v.next.append(CreateTree(A[n // 2 :, n // 2 :],
                        r, eps, sMin + n // 2, tMin + n // 2))
    return v
```

Funkcja `Decompress()` odpowiada za dekompresje macierzy z uzyskanej struktury drzewiastej.

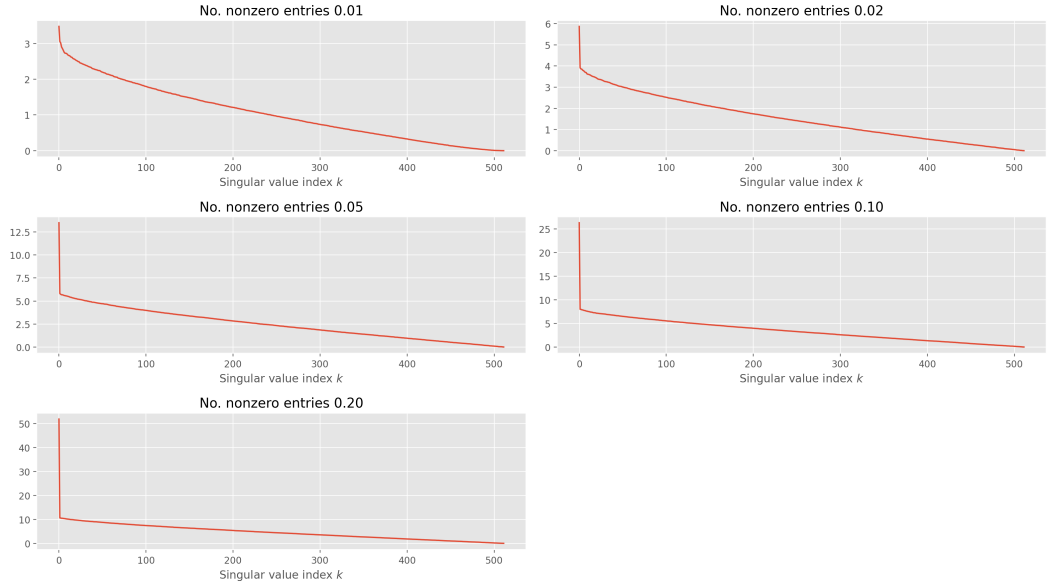
```
def Decompress(root: Node):
    n = root.side
    A = np.zeros((n, n))

    def DecompressRecursive(root: Node):
        nonlocal A
        if root.rank is None:  # Not a leaf -> pass
            pass
        elif root.rank == 0:  # Leaf with all 0s -> pass
            pass
        elif root.rank > 0:
            s, t, side = root.sMin, root.tMin, root.side
            U, D, V = root.U, np.diag(root.D), root.V
            A[s : s + side, t : t + side] = U @ D @ V

        for node in root.next:
            DecompressRecursive(node)

    DecompressRecursive(root)
    return A
```

Analogicznie została zaimplementowana funkcja służąca do wizualizacji.

```
def DrawTree(root: Node):
    n = root.side
    fig, ax = plt.subplots()
    ax.set_aspect("equal", "box")
    ax.set_xlim(0, n)
    ax.set_ylim(0, n)

    def DrawTreeRecursive(root: Node):
        nonlocal ax

        if root.rank is None:  # Not a leaf -> draw grid lines
            x, y, d = root.sMin, root.tMin, root.side // 2
            ax.plot((x, x + 2 * d), (y + d, y + d), color="k", lw=0.6)
            ax.plot((x + d, x + d), (y, y + 2 * d), color="k", lw=0.6)

        elif root.rank > 0:  # Leaf with SVD -> fill block
            x, y, d = root.sMin, root.tMin + root.side, root.side
            lw = root.side / (6 * root.rank)
            for i in range(root.rank):
                ax.fill([x, x + d, x + d, x],
                        [y, y, y - (i + 1) * lw, y - (i + 1) * lw],
                        color="k")
                ax.fill([x, x + (i + 1) * lw, x + (i + 1) * lw, x],
                        [y, y, y - d, y - d],
                        color="k")

        elif root.rank == 0:  # Leaf with all 0s -> pass
            pass

        for node in root.next:
            DrawTreeRecursive(node)

    DrawTreeRecursive(root)
    fig.show()
```

3 Wyniki

Wylosowano macierze wymiarów 512×512 zawierające odpowiednio 1, 2, 5, 10, i 20 procent elementów niezerowych. Dla każdej z macierzy wykonano wykres wartości osobliwych, który zamieszczono poniżej.

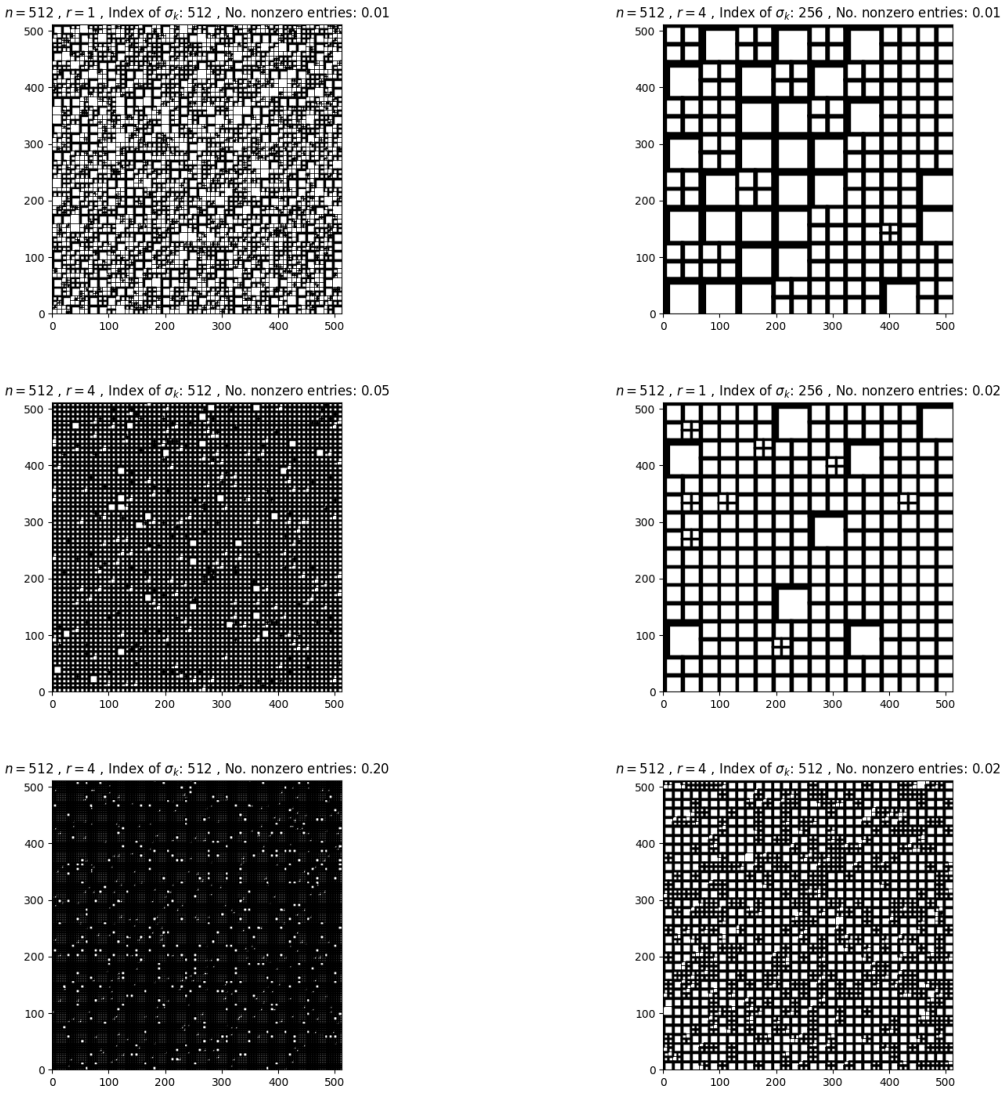


Rysunek 1: Wykres wartości osobliwych dla różnych losowych macierzy

Następnie dla wylosowanych macierzy oraz parametrów r (maksymalny rank) oraz ϵ (minimalna wartość osobliwa) ze zbiorów odpowiednio $\{1, 4\}$ i $\{\sigma_2, \sigma_{2^{1023}}, \sigma 2^{1024}\}$ wykonano kompresje macierzy oraz zwizualizowano powstałe w ten sposób macierze hierarchiczne. Dodatkowo zmierzono czasy kompresji. Uzyskane wyniki zawarto poniżej. W Tabeli 1 zamieszczono zmierzone czasy kompresji oraz obliczony błąd średniokwadratowy. Względny błąd średniokwadratowy został obliczony jako stosunek

$$\frac{\left\|\mathbf{A} - \tilde{\mathbf{A}}\right\|}{\|\mathbf{A}\|} = \sqrt{\frac{\sum_{i,j}(A_{ij} - \tilde{A}_{ij})^2}{\sum_{i,j} A_{ij}^2}}, \tag{1}$$

gdzie $\mathbf{A}$ jest macierzą przed kompresją, a $\tilde{\mathbf{A}}$ jest macierzą $\mathbf{A}$ po dekompresji ze struktury macierzy hierarchicznej. Na Rysunku 2 zamieszczono również wizualizacje otrzymanych macierzy hierarchicznych.



Rysunek 2: Wizualizacje uzyskanych macierzy hierarchicznych

Proc. zerowych wartości	r	indeks σ_k	Czas [s]	Błąd średniokwadratowy
0.99	1	512	1.07	1.45e-16
0.99	1	2	0.01	9.88e-01
0.99	1	256	0.26	5.60e-01
0.99	4	512	0.53	7.35e-16
0.99	4	2	0.01	9.74e-01
0.99	4	256	0.11	5.02e-01
0.98	1	512	1.89	1.50e-16
0.98	1	2	0.01	9.86e-01
0.98	1	256	0.17	7.78e-01
0.98	4	512	0.69	7.58e-16
0.98	4	2	0.01	9.73e-01
0.98	4	256	0.09	7.39e-01
0.95	1	512	3.77	1.51e-16
0.95	1	2	0.01	9.75e-01
0.95	1	256	0.09	8.97e-01
0.95	4	512	1.18	8.55e-16
0.95	4	2	0.01	9.64e-01
0.95	4	256	0.08	8.17e-01
0.90	1	512	6.50	1.54e-16
0.90	1	2	0.02	9.56e-01
0.90	1	256	0.07	9.06e-01
0.90	4	512	1.83	9.17e-16
0.90	4	2	0.01	9.45e-01
0.90	4	256	0.07	8.21e-01
0.80	1	512	10.45	1.60e-16
0.80	1	2	0.01	9.17e-01
0.80	1	256	0.07	8.77e-01
0.80	4	512	3.03	7.60e-16
0.80	4	2	0.02	9.06e-01
0.80	4	256	0.08	7.99e-01

Tabela 1: Tabela porównująca czas kompresji i błąd dla różnych macierzy i parametrów r, ϵ