Raport

rekurencyjna kompresja macierzy z wykorzystaniem częściowego SVD

Szymon Twardosz, Dominik Jeżów

20 grudnia 2023

1 Środowisko

Do wykonania ćwiczenia wykorzystaliśmy język python 3.11 wraz z następującymi bibliotekami numpy, matplotlib, sklearn oraz time

2 Temat zadania

Zadanie polegało na zaimplementowaniu i przetestowaniu rekurencyjnej metody korzystającej z częściowego SVD. Dodatkowo, należało stworzyć narzędzie umożliwiające wizualizację skompresowanej macierzy.

Proces testowania obejmował generowanie serii 5-elementowych dużych macierzy z różnym stopniem zagęszczenia, kolejno: 1%, 2%, 5%, 10%, 20%. Następnie, po wygenerowaniu każdej macierzy, mierzone były czasy kompresji SVD dla dwóch różnych dopuszczalnych rzędów macierzy.

Dodatkowo, należało zaimplementować funkcję dekompresującą i ocenić, jak bardzo różni się oryginalna macierz od zdekompresowanej.

3 Pseudokod

```
CreateTree(tmin, tmax, smin, smax, r, epsilon):
    U, D, V = truncatedSVD(A(tmin:tmax ; smin:smax))
    if D[r+1, r+1] < epsilon:
        v = CompressMatrix(tmin, tmax, smin, smax, U, D, V, r)
    else:
        v = new node
        v.append(CreateTree(tmin, tnewmax, smin, snewmax))
        v.append(CreateTree(tmin, tnewmax, snewmax+1, smax))
        v.append(CreateTree(tnewmax+1, tmax, smin, snewmax))
        v.append(CreateTree(tnewmax+1, tmax, snewmax+1, smax))
    return v
```

Jak widać w powyższym pseudokodzie, pierwszym krokiem jest próba skompresowania pewnego fragmentu macierzy za pomocą techniki SVD. Następnie, w zależności od jakości uzyskanej kompresji, podejmujemy decyzję, czy zachować tę kompresję, czy też dokonać kompresji na cztery mniejsze fragmenty.

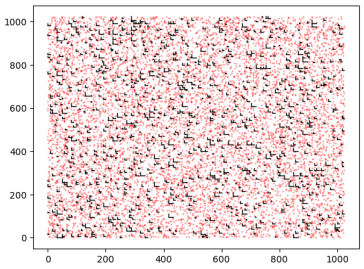
4 Ważniejsze fragmenty kodu

```
def create_tree(matrix, r, epsilon):
U, s, V = randomized_svd(matrix, n_components=r)

node = None
if s[-1] < epsilon:
    node = compress_matrix(matrix, U, s, V, r)

else:
    Y, X = matrix.shape

    if Y == X == 1: # cannot segment matrix
        node = CompressNode(rank=1, size=(1, 1))
        node.val = matrix
        return node

    node = CompressNode(rank=r, children=[], size=(Y, X))

    # 4 childrens

    node.append_child(create_tree(matrix[0: Y // 2, 0: X // 2], r, epsilon))
    node.append_child(create_tree(matrix[0: Y // 2, X // 2: X], r, epsilon))
    node.append_child(create_tree(matrix[Y // 2: Y, 0: X // 2], r, epsilon))
    node.append_child(create_tree(matrix[Y // 2: Y, X // 2: X], r, epsilon))

return node
```

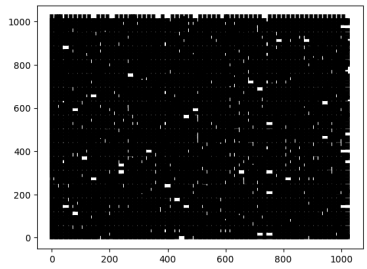
W naszej implementacji obliczamy SVD korzystając z biblioteki scikit-learn (sklearn). Poza dodaniem warunku końcowego, obsługującego macierz wejściową o wymiarach 1x1, nasza implementacja nie różni się znacząco od pseudokodu.

5 Wyniki

W związku z długim czasem działania pętli testującej, który wynosi około 200 minut, udało nam się zebrać wyniki tylko dla gęstości do 5%.

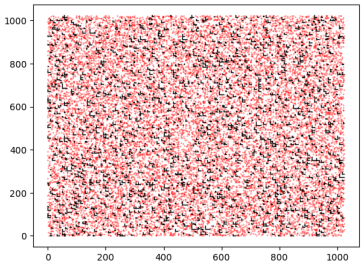


(a) makymalny rząd równy 1

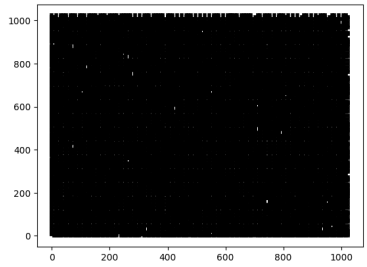


(b) makymalny rząd równy 4

Rysunek 1: gęstości równa 1 procent



(a) makymalny rząd równy 1

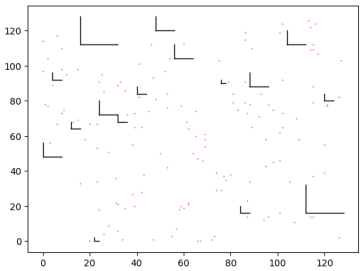


(b) makymalny rząd równy 4

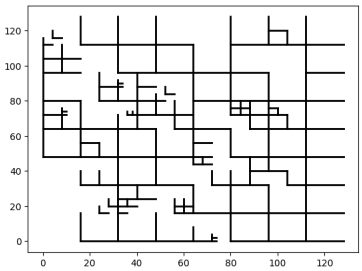
Rysunek 2: gęstości równa 2 procent

5.1 wizualizacje wybranych macierzy

Na rysunkach 1 i 2 można zauważyć, że dla niższego rzędu występują czerwone punkty. Oznaczają one zatrzymanie algorytmu na końcowym warunku dla macierzy o rozmiarze 1x1. Obserwuje się również dużą ilość małych kompresji SVD.



(a) makymalny rząd równy 1



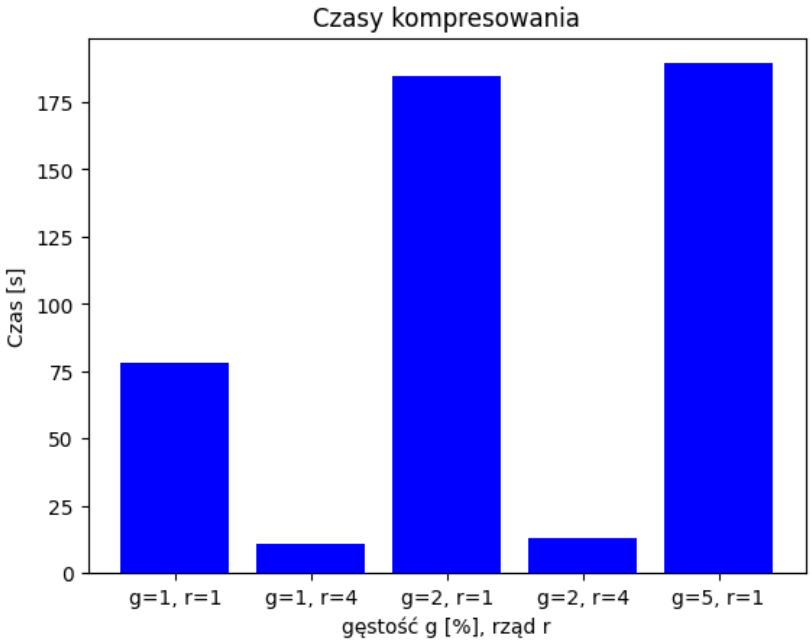
(b) makymalny rząd równy 2

Rysunek 3: Dobrze zrobione SVD

Na rysunku 3, w przeciwieństwie do poprzednich, wielkość poszczególnych kompresji jest znaczna względem całej długości boku. Dane wejściowe dla rysunków 1, 2 oraz 3 różnią się wielkością macierzy; w pierwszych dwóch przypadkach jest ona 8 razy większa.

5.2 Czasy kompresji

Jak widać na wykresie 4, zwiększenie dopuszczalnego rzędu macierzy znacznie skraca czas kompresji. Obserwuje się również tendencję wzrostową w zagęszczaniu macierzy od upływem czasu.

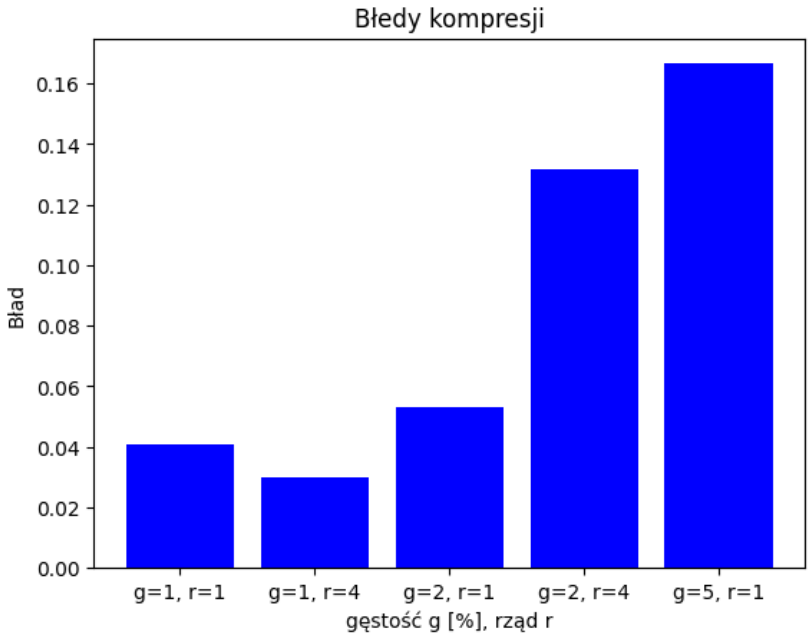


Rysunek 4: Wykres słupkowy rodzaju kompresji od czasu

5.3 Błędy stratności kompresji

Widoczne jest, że dla przyjętego epsilonu otrzymywane błędy są znaczące, sięgające nawet 0.16. Biorąc pod uwagę, że wartości w macierzach znajdują się w przedziale ¡0,1¿, wydaje się, że ta forma kompresji jest kiepskim pomysłem.

Otrzymane wyniki są prawdopodobnie spowodowane zbyt dużą wartością epsilonu, której dobór był zbyt wysoki w celu przyspieszenia obliczeń.



Rysunek 5: Wykres słupkowy rodzaju kompresji od błędu kwadratowego