Normy macierzowe i rozkład według wartości osobliwych (SVD)

Rachunek Macierzowy i Statystyka Wielowymiarowa

Ewa Pelc
Ewa Żukowska

Dane

Do testów poprawności funkcji wykorzystano macierz tybetańską rozmiaru 3x3 z wykładu.

```
a1 = np.array([[4, 9, 2], [3, 5, 7], [8, 1, 6]])
```

```
array([[4, 9, 2],
       [3, 5, 7],
       [8, 1, 6]])
```

Norma indukowana jedynkowa

Norma indukowana jedynkowa jest obliczona jako maksymalna suma wartości bezwzględnych z kolumn. Jest to alternatywny sposób obliczania normy jednostkowej zaprezentowany na wykładzie.

W celu ułatwienia sumowania w bibliotece numpy, najpierw wykonano transpozycję macierzy - aby móc sumować w kolumnach.

```
def norm1(m):
    a = m.T
    return np.max([sum(abs(row))for row in a])
```

Wynik dla macierzy a1:

```
norm1(a1)
```

```
15
```

Współczynnik uwarunkowania dla normy jedynkowej i macierzy a1:

Do obliczenia macierzy odwrotnej posłużono się funkcją biblioteki numpy

```
reversed = np.linalg.inv(a1)
```

```
reversed
```

```
array([[ 0.06388889, -0.14444444,  0.14722222],
       [ 0.10555556,  0.02222222, -0.06111111],
       [-0.10277778,  0.18888889, -0.01944444]])
```

```
norm1(a1) * norm1(reversed)
```

```
5.333333333333333
```

Interpretacja współczynnika uwarunkowania

Współczynnik uwarunkowania macierzy mówi nam, jak bardzo dane rozwiązanie jest wrażliwe na niewielkie zmiany wejściowe, takie jak zaburzenia w danych lub błędy numeryczne.

Im większy współczynnik uwarunkowania, tym bardziej macierz jest źle uwarunkowana, co oznacza, że jest bardziej wrażliwa na zmiany

Norma indukowana nieskończoność

Norma indukowana nieskończoność jest obliczona jako maksymalna suma wartości bezwzględnych z wierszy. Jest to alternatywny sposób obliczania normy nieskończoność zaprezentowany na wykładzie.

```
def norm_inf(m):
    return np.max([sum(abs(row)) for row in m])
```

Wynik dla macierzy a1:

```
norm_inf(a1)
```

```
15
```

Współczynnik uwarunkowania:

```
norm_inf(a1) * norm_inf(reversed)
```

```
5.333333333333333
```

Norma indukowana dwójkowa (norma spektralna)

Norma indukowana dwójkowa jest obliczana największa (na moduł) wartość własna macierzy A. Jest to alternatywny sposób obliczania zaprezentowany na wykładzie.

Zgodnie z wytycznymi, do wyznaczenia wektorów i wartości własnych macierzy posłużono się biblioteką numerycznej z wybranego języka programowania.

```
def norm2(m):
    return np.max(abs(np.linalg.eig(m).eigenvalues))
```

Wynik dla macierzy a1:

```
norm2(a1)
```

```
15.000000000000002
```

Współczynnik uwarunkowania:

```
norm2(a1) * norm2(reversed)
```

```
3.0618621784789704
```

p-norma macierzowa

P-norma macierzowa jest obliczana jako p-norma wektorowa zastosowana do każdego wiersza macierzy, a następnie obliczenie sumy wyników. W rezultacie otrzymujemy sumę wszystkich elementów podniesionych do potęgi p, a następnie obliczamy pierwiastek p-tego stopnia z tej sumy.

Wykonano obliczenia dla p=4

```
def normp(m, p):
    inner = np.sum(np.power(m, p))
    outer = np.power(inner, (1/p))
    return outer
```

Wynik dla macierzy a1:

```
normp(a1, 4)
```

```
11.127735237293699
```

Współczynnik uwarunkowania:

```
normp(a1, 4) * normp(reversed, 4)
```

```
2.4743886469620096
```

Rozkład według wartości własnych - Singular Value Decomposition (SVD)

W ramach projektu przeprowadzono implementację algorytmu Singular Value Decomposition (SVD) do rozkładu macierzy.

Każdą macierz rzeczywistą A można przedstawić w postaci rozkładu SVD:

$$A = U\Sigma V^T,$$

gdzie:

- U i V – macierze ortogonalne (czyli $U^{-1} = U^T$, $V^{-1} = V^T$),
- Σ – macierz diagonalna (przekątniowa), taka że $\Sigma = \mathrm{diag}(\sigma_i)$, gdzie σ_i – nieujemne wartości szczególne (osobliwe) macierzy A, zwyczajowo uporządkowane nierosnąco.

Pseudokod

> Input: Macierz A o wymiarach m x n
> Output: Macierze U, ∑, V^T
>
> 1. Oblicz macierz kowariancji C = A^T * A
> 2. Znajdź dominujące wektory własne macierzy kowariancji C za pomocą metody potęgowej:
>   Dla każdego i od 1 do k:
>   a) Losowo wygeneruj wektor startowy v
>   b) Dla zadanego kroku iteracyjnego j od 1 do max_iter: v = C * v v = v / ||v||
>   c) Dodaj uzyskany wektor własny do listy dominujących wektorów własnych
> 3. Przekształć uzyskane dominujące wektory własne na macierze ortogonalne U i V: U = [v_1, v_2, ..., v_k] V = [u_1, u_2, ..., u_k]
> 4. Oblicz wartości własne macierzy kowariancji C i przekształć je w macierz diagonalną ∑: Dla każdej wartości własnej λ_i: ∑_i = sqrt(λ_i) ∑ = diag([∑_1, ∑_2, ..., ∑_k])
> 5. Zwróć macierze U, ∑, V transponowane

Kod

```
def svd(A):
    # Obliczenie macierzy kowariancji
    C = np.dot(A.T, A)

    # Znalezienie dominujących wektorów własnych
    # macierzy kowariancji
    eigenvalues, eigenvectors = np.linalg.eigh(C)

    # Przekształcenie dominujących wektorów własnych
    # na macierze ortogonalne U i V
    U = eigenvectors
    V = np.dot(A, U) / np.linalg.norm(np.dot(A, U), axis=0)

    # Obliczenie macierzy Sigma
    s = np.sqrt(np.abs(eigenvalues))
    Sigma = np.diag(s)
```

```
    return U, Sigma, V.T
```

Wyniki

```
A = np.array([[1, 2], [3, 4], [5, 6]])
U, Sigma, Vt = svd(A)
```

Macierz U:

```
[-0.78489445  0.61962948]
[ 0.61962948  0.78489445]
```

Macierz Sigma:

```
[0.51430058 0.        ]
[0.         9.52551809]
```

Macierz V transponowane:

```
[ 0.88346102  0.24078249 -0.40189603]
[ 0.2298477   0.52474482  0.81964194]
```

Macierz UΣV^T = A:

```
[1. 3. 5.]
[2. 4. 6.]
```