Eliminacja Gaussa i Faktoryzacja LU

Rachunek Macierzowy i Statystyka Wielowymiarowa

Ewa Pelc
Ewa Żukowska

Faktoryzacja LU

Rozmiar macierzy: 20 + 5 = 25

```
n <- 25
A <- matrix(runif(n^2), nrow = n)
```

Pseudokod algorytmu LU

1. Zainicjuj $\mathbf{L}$ jako macierz identycznościową, $\mathbf{I}$ o wymiarach $n \times n$ oraz $\mathbf{U} = \mathbf{A}$.
2. Dla $i = 1$,…, n wykonaj krok 3.
3. Dla $j = i + 1$, …, n wykonaj kroki 4-5.
4. Ustaw $lji=uji/uii$.
5. Wykonaj $Uj=(Uj - lji * Ui)$ (gdzie Ui, Uj reprezentują odpowiednio wiersze i oraz j macierzy $\mathbf{U}$).

Algorytm faktoryzacji LU:

```
LU_factorization <- function(A) {
  n <- nrow(A)
  L <- matrix(0, nrow = n, ncol = n)
  U <- matrix(0, nrow = n, ncol = n)
  print(A)

  for (i in 1:n) {
    for (j in i:n) {
      U[i, j] <- A[i,j] - L[i,1:(i-1)] %*% U[1:(i-1),j]
    }
    for (j in i:n) {
      L[j,i] <- (A[j,i] - L[j,1:(i-1)] %*% U[1:(i-1),i]) / U[i,i]
    }
    L[i, i] <- 1
  }

  return(list(L = L, U = U))
}
```

Sprawdzenie poprawności LU faktoryzacji

W celu sprawdzenia poprawności faktoryzacji zdefiniowałyśmy funkcję, która porównuje dwa obiekty dopuszczając ustalony błąd.

```
is_allclose <- function(a, b, tol = 1e-10) {
  max_diff <- max(abs(a - b))
  return(max_diff < tol)
```

```
}
```

Sprawdzenie, czy A = LU

```
if (is_allclose(A, L %*% U)) {
  print("LU faktoryzacja jest poprawna.")
} else {
  print("LU faktoryzacja jest niepoprawna.")
}
```

Pseudokod algorytmu LU z pivotingiem

1. Zainicjuj $\mathbf{L}$ jako macierz identycznościową, $\mathbf{I}$ o wymiarach $n \times n$ oraz $\mathbf{U} = \mathbf{A}$.
2. Zainicjuj wektor permutacji P na [1, 2, ..., n].
3. Dla k = 1, …, n wykonaj kroki 4-13.
4. Ustaw l_max$k$ = k oraz e_max$k$ = | $A[P[k], k]$ |.
5. Dla $i = k + 1, \ldots, n$ wykonaj krok 6.
6. Jeśli |$A[P[i], k]$| > e_max$k$, to ustaw l_max$k$ = i oraz e_max$k$ = |$A[P[i], k]$|.
7. Jeśli e_max$k$ ≤ ε, to zakończ z wynikiem "false" (macierz jest zdegenerowana).
8. Jeśli l_max$k$ ≠ k, to zamień miejscami $P[k]$ i $P[l$max$k]$.
9. Jeśli $U[P[k], k] \neq 0$, to wykonaj krok 10, w przeciwnym razie przesuń się do kolejnego kroku.
10. Ustaw $ukk = A[P[k], k]$.
11. Dla $i = k+1, \ldots, n$ wykonaj: $A[P[i], k] \leftarrow A[P[i], k] / ukk$.
12. Dla $i = k+1, \ldots, n$ wykonaj kroki 13.
13. Dla $j = k+1, \ldots, n$ wykonaj: $A[P[i], j] \leftarrow A[P[i], j] - A[P[i], k] * A[P[k], j]$.
14. Zakończ z wynikiem "true" i zwróć L, U, P.

Algorytm faktoryzacji LU z pivotingiem:

```
LU_factorization_pivot <- function(A, epsilon = 1e-10) {
  n <- nrow(A)
  md <- 1
  W <- 1:n
  L <- diag(1, nrow = n)
  U <- matrix(0, nrow = n, ncol = n)

  for (k in 1:(n-1)) {
    maxw <- k
    maxe <- abs(A[W[k], k])

    for (i in (k+1):n) {
      if (abs(A[W[i], k]) > maxe) {
        maxw <- i
        maxe <- abs(A[W[i], k])
      }
    }
```

```
    if (maxe <= epsilon) {
      return(NULL)  # Macierz jest zdegenerowana
    }

    if (maxw != W[k]) {
      md <- -md
      temp <- W[k]
      W[k] <- W[maxw]
      W[maxw] <- temp
    }

    U[k, ] <- A[W[k], ]

    for (i in (k+1):n) {
      L[i, k] <- A[W[i], k] / A[W[k], k]
      U[i, ] <- A[W[i], ] - L[i, k] * U[k, ]
    }
  }

  U[n, ] <- A[W[n], ]

  return(list(L = L, U = U))
}
```

```
LU_decomposition_with_pivoting <- LU_factorization_pivot(A)
L <- LU_decomposition_with_pivoting$L
U <- LU_decomposition_with_pivoting$U
```

```
LU_factorization_pivot(A)
```

Sprawdzenie, czy A = PLU:

```
if (is_allclose(A, P %*% L %*% U)) {
  print("LU faktoryzacja z pivotingiem jest poprawna.")
} else {
  print("LU faktoryzacja z pivotingiem jest niepoprawna.")
}
```