Computational Laplace Expansion

Jason Zhang

May 29, 2022

1 Laplace Expansion

In linear algebra, Laplace expansion, also called cofactor expansion, is one of many mathematical algorithms to express the determinant of a matrix. Specifically, the Laplace expansion expresses the determinant as a linear combination of determinants of smaller matrices, namely minors. This expansion is recursive by nature, however, for larger matrices, the determinant becomes inefficient to compute.

1.1 Minors

A complementary minor is the determinant of a sub-matrix with a designated row and column of the original matrix A eliminated.

Definition: Complementary Minor

Let matrix A be $n \times n$ (where $n \geq 2$) and A_{ij} is the entry of A at the i^{th} row and j^{th} column. The sub-matrix is obtained from the original matrix A by deleting the respective i^{th} row and j^{th} column. Then the ***complementary minor*** *of A_{ij} is the determinant of the sub-matrix.*

1.2 Cofactors

A cofactor is a coefficient that results the complementary minor of A_{ij} a positive or negative value. This positive or negative determination is dependent upon the location of the entry A_{ij} in reference to the original matrix A.

Definition: Cofactors of a Matrix

Suppose that determinants of $(n-1)\times(n-1)$ matrices are found. Given the $n \times n$ matrix A, let B_{ij} denote the $(n-1)\times(n-1)$ matrix obtained from A by deleting the i^{th} row and j^{th} column.

*Then the (i, j)-**cofactor** $c_{ij}(A)$ is the scalar defined by*

$$c_{ij}(A) = (-1)^{i+j} \det(A_{ij})$$

*Here $(-1)^{i+j}$ is called the **sign** of the (i, j)-position.*

To visualize, the pattern of sign changes $(-1)^{i+j}$ of an $n \times n$ is:

$$\begin{bmatrix} + & - & + & - & \dots \\ - & + & - & + & \dots \\ + & - & + & - & \dots \\ - & + & - & + & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

Note that the signs are either 1 or -1 and alternate across each row and column.

1.3 Expansion

Given the previous two definitions, we can now define det A for any square matrix A.

Definition: Laplace Expansion of a Matrix

Suppose that determinants of $(n-1)\times(n-1)$ *sub-matrices of the original matrix* A *are found. If* $A = [a_{ij}]$ *is* $n \times n$ *(where* $n \geq 2$*) define*

$$\det A = a_{11}c_{11}(A) + a_{12}c_{12}(A) + \ldots + a_{1n}c_{1n}(A)$$

Then for any row i*, generalize*

$$\det A = \sum_{i=1}^{n} a_{ij}c_{ij}$$

The above expression is the ***Laplace (cofactor) expansion*** *of* det A *along row* 1.

The above definition expanded along row 1. However, the det A can calculated by expanding along any row or column.

Theorem: Laplace Expansion Theorem

The determinant of an $n \times n$ *matrix* A *(where* $n \geq 2$*) can be computed by using the Laplace expansion along any row or column of* A*. In other words,* det A *can be computed by multiplying each entry of the row or column with corresponding cofactors and summing the results.*

1.4 Example

Example: Laplace Expansion

Given the matrix $A = \begin{bmatrix} 1 & 3 & -6 \\ 3 & 4 & 1 \\ 9 & 5 & 2 \end{bmatrix}$ *compute the determinant.*

By the previously stated theorem, we can choose any row or column to expand upon. In this example, we arbitrarily select the first row to conduct the Laplace expansion. The reader is welcomed to verify that det A can be computed by expanding across any row or column.

By the definition of the Laplace Expansion of a Matrix:

$$\begin{aligned}\det A &= a_{11}c_{11}(A) + a_{12}c_{12}(A) + a_{13}c_{13}(A)\\ &= 1c_{11}(A) + 3c_{12}(A) - 6c_{13}(A)\\ &= 1(-1)^{1+1}\begin{vmatrix} 4 & 1 \\ 5 & 2 \end{vmatrix} + 3(-1)^{1+2}\begin{vmatrix} 3 & 1 \\ 9 & 2 \end{vmatrix} - 6(-1)^{1+3}\begin{vmatrix} 3 & 4 \\ 9 & 5 \end{vmatrix}\\ &= 138\end{aligned}$$

To remind, the three sub-matrices are obtained by deleting the i^{th} row and j^{th} column. Below illustrates the deletion of the A_{11} entry to obtain the encircled first sub-matrix:

$$\begin{bmatrix} 1 & 3 & -6 \\ 3 & 4 & 1 \\ 9 & 5 & 2 \end{bmatrix}$$

2 Code Functionality

The program written in the C computer programming language functions the Laplace Expansion algorithm. The complete code is linked here. The program prompts the square matrix size n and entries A_{ij}. Dynamic memory allocation is used to store matrix entries and complementary minors. The Laplace Expansion algorithm is implemented by recursion. The concerned block of code (written in C) is provided below:

```
    for (originalCol = 0; originalCol < matrixSize; originalCol++){
        for (row = 1; row < matrixSize; row++){
            for (col = 0; col < originalCol; col++){
                subMatrix[row - 1][col] = matrix[row][col];
            }
            for (col = originalCol + 1; col < matrixSize; col++){
                subMatrix[row - 1][col - 1] = matrix[row][col];
            }
        }

        int recurDet = laplaceExpand(matrixSize - 1, subMatrix, guard);

        if (guard == NULL){
            free(subMatrix);
            return 0;
        }

        totalDet += sign * matrix[0][originalCol] * recurDet;
        sign *= -1;
    }
```

2.1 Copying Complementary Minors

The first part of the code involves copying the complementary minor entries from the original matrix, and storing the copied entries into a smaller sub-matrix, specifically lines 1-9. To provide context, `largerCol` is an empty 2-D array with the original matrix size minus one, dynamically allocated earlier in the program.

```
1     for (largerCol = 0; largerCol < matrixSize; largerCol++){
2         for (row = 1; row < matrixSize; row++){
3             for (col = 0; col < largerCol; col++){
4                 subMatrix[row - 1][col] = matrix[row][col];
5             }
6             for (col = largerCol + 1; col < matrixSize; col++){
7                 subMatrix[row - 1][col - 1] = matrix[row][col];
8             }
9         }
```

By the Definition: Complementary Minor, a sub-matrix is obtained from the original matrix by deleting the i^{th} row and j^{th} column.

> To establish the j^{th} column, the first for loop (line 1) with index `largerCol`, iterates through each column of the matrix. Thus, `largerCol` is the j^{th} column.
>
> To establish the i^{th} row, the Laplace Expansion Theorem states that, one can choose any row or column to expand upon. In the code, row 1 is arbitrarily selected. To do this, let the second for loop's (line 2) initial index, `row = 1`, such that the second for loop starts iterating from the matrix's second row; completely ignoring the matrix's first row. Thus, row 1 is the i^{th} row.

With the i^{th} row and j^{th} columns established, the next step is to copy the entries (that are not a part of the i^{th} row and j^{th} columns) and store them into their respective sub-matrices.

> The second for loop iterates through each row, starting from the matrix's second row. The two remaining inner third tier for loops (line 3 and 6) iterate and copy the entries left and right of the j^{th} column, `largerCol`, respectively. It can be seen that for each iteration of the first for loop, `largerCol` effectively divides the matrix into 2 parts.

As an example, below illustrates a 3×3 matrix, where row 1 is the i^{th} row and the different iterations of `largerCol`, namely the j^{th} column. The entries not highlighted by the i^{th} row or the j^{th} column are the entries that constitute the sub-matrix.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

2.2 Recursion Process

The second part of the code involves repeating the first part by recursion, in lines 10-18. Specifically, if `subMatrix` is not a 2×2 matrix, `subMatrix` will repeatedly undergo the first part of the code until it is reduced to a 2×2 matrix. Once reduced to a 2×2 matrix, then a simple calculation will

yield the determinant for that specific sub-matrix. With the collection of sub-matrices, the sum of their determinants along with the multiplied sign and entry yield the determinant of the original matrix in question.

```
10        int recurDet = laplaceExpand(matrixSize - 1, subMatrix, guard);
11
12        if (guard == NULL){
13            free(subMatrix);
14            return 0;
15        }

16        totalDet += sign * matrix[0][originalCol] * recurDet;
17        sign *= -1;
18    }
```

Call by recursion occurs in line 10, where the function is named `laplaceExpand`. A base case exists earlier in the code such that line 10 no longer executes once `subMatrix` is a 2×2 matrix. The base case calculates and returns the determinant of the 2×2 `subMatrix`.

The guard condition in lines 12-15 ensures that sufficient memory is allocated for `subMatrix`.

Line 16-17 highlights the calculation portion of Definition: Laplace Expansion of a Matrix.

> The `totalDet` variable stores the sum of the Laplace Expansion.
>
> In reference to the Definition: Cofactors of a Matrix, the sign of a matrix alternates between -1 and 1. Thus, line 17 facilitates this alternation with each calculation.
>
> The second term, `matrix[0][originalCol]` is the entry of the i^{th} row and j^{th} column. The i^{th} row is 0 because the code has arbitrarily established the i^{th} row as the first row of the matrix. The j^{th} column is the `largerCol` variable.
>
> The last term, `recurDet` is the determinant of the 2×2 `subMatrix`.

3 Computational Expense

The time complexity to conduct the Laplace Expansion algorithm is $\mathcal{O}(n!)$. To explain, the determinant calculation of a 2×2 matrix comprise 2 multiplications and 1 addition. Thus, 3 operations. Now applying the Laplace Expansion for a 3×3 matrix, there consists three 2×2 sub-matrices. Thus, 3 times 3 operations. Applying the Laplace Expansion for a 4×4 matrix, there consists four 3×3 sub-matrices, and in each of those 3×3 sub-matrices, there consists three 2×2 sub-matrices. Thus, 4 times 3 times 3 operations. $\mathcal{O}(n!)$ approximates this mathematical pattern of increasing matrix sizes, where n is the matrix size. Note that a formal proof exists that validates this time complexity.