Determinant Calculation in R

Rahul Goswami

2022-01-17

This is a recursive function that implements determinant of a matrix. It takes as input a matrix and spits out determinant of the matrix, however it is not efficient since it's way too slow. when i had written this code for the first time i was very happy with the result. but when i came to know about linear algebra in R, i was very disappointed. since this function will take years to calculate the determinant of a 20x20 matrix.

```r
r.det<-function(x)
{
  # a stores the number of rows and columns of the matrix
  a<-dim(x);
  # if the matrix is 1x1, then return the value of the matrix
  if(a[1] == 1 && a[2] == 1)
    return(x[1,1])
  # if the matrix is 2x2, then return the value of the determinant
  if(a[1]==2 && a[2]==2)
    return(x[1,1]*x[2,2]-x[1,2]*x[2,1])
  # if the matrix dimension is greater than 2x2 , then it will calculate Co-factors and then
  # and calls itself recusively
  else
  {
    det<-0
  for(i in 1:a[1])
  {
    det<-det+(-1)^(1+i)*x[1,i]*r.det(x[-1,-i])
  }
  }
  return(det);
}
```

This function looks cool works great on small matrices, let us see an Example ### Example Take A matrix

```r
vector = 1:9 # A vector of size 9, which we will use to create a matrix
A = matrix(vector,3,3)  # A 3x3 matrix
deteminant_of_A = r.det(A) # The determinant of A
```

we know the determinant of A is 0, as calculated by the function r.det(A) Now let us see how it works on a bigger matrix, with dimension 20x20

```r
Big_Matrix = matrix(sample.int(20, size = 150, replace = TRUE), nrow = 10, ncol = 10)
```

The determinant of the matrix is calculated by the function r.det(Big_Matrix) Let us check the time taken for the function to run

```r
start.time <- Sys.time()
deteminant_of_Big_Matrix = r.det(Big_Matrix)
end.time <- Sys.time()
```

```r
cat("Time taken for the function to run is ", end.time - start.time, " seconds")
```

```
## Time taken for the function to run is  7.267692  seconds
```

The time taken for the function to run is too much and considering that in Machine Learning we are dealing with large matrices, we will not be able to calculate the determinant of a matrix in a reasonable time. so I searched through some Linear Algebra books and gets to know for cause of this and a solution.

Let's Talk about the Cause

The determinant of a matrix is calculated by the trivial formula takes a lot of time to calulate and it is not efficient for large matrices. It happens because the number of operations computer has to perform is too much. The number of operations computer is of O(n!) where n is the dimension of the matrix.

Solution to the problem

The solution to the problem is to convert the matrix into a lower triangular matrix or upper triangular matrix. The lower triangular matrix is a matrix where all the elements below the main diagonal are zero. The upper triangular matrix is a matrix where all the elements above the main diagonal are zero and the determinant is just the product of the diagonal elements. The number of operations step is O(n^3) where n is the dimension of the matrix. For details check the Gilbert Strang's book Linear Algebra and its applications. Now let us write a function to calculate the determinant of a matrix in a reasonable time named r.fast.det

```r
r.fast.det <- function(A){
  # A function to Convert the matrix into a lower triangular matrix
  lower_triangular<-function(x)
  {
    if(is.matrix(x)==1)
    {
      for(i in 2:nrow(x))
      {
        x[i, ]<-x[i, ]-x[i,1]*(x[1, ]/x[1,1])
      }
      x[-1,-1]<-lower_triangular(x[-1,-1])
    }
  x
  }

  lower_triangular_A = lower_triangular(A)
  # Determinant is product of diagonal elements of lower_triangular_A
  determinant = 1
  for(i in 1:nrow(lower_triangular_A))
  {
    determinant<-determinant*lower_triangular_A[i,i]
  }

  determinant
}
```

The determinant of the matrix is calculated by the function r.fast.det(Big_Matrix) Let us check the time taken for the function to run

```r
start.time <- Sys.time()
deteminant_of_Big_Matrix = r.fast.det(Big_Matrix)
end.time <- Sys.time()
cat("Time taken for the function to run is ", end.time - start.time, " seconds")
```

```
## Time taken for the function to run is  0.02598786  seconds
```