矩阵乘法中的分治法

19222126 计科221 董自经

一、传统矩阵乘法

传统的矩阵乘法是使用矩阵的行和列进行直接计算，以A=（aij）m\*s和B=(bij)s\*n为例，A与B的乘积定义为

Cm\*n=AB= c11 c12 … c1n

c21 c22 … c2n

. . . .

cm1 cm2 … cmn

，其中Cij=ai1b1j+ai2b2j+…+aisbsj=（i=1,2,…,m; j=1,2,…,n）。在矩阵相乘前需要判断前一个矩阵的列数等于后一个矩阵的行数，在此默认输入的矩阵满足此条件，不做判断。

对于矩阵乘法，其中还有很多的特殊情况，比如对角矩阵的自乘，矩阵和单位阵相乘，矩形和调整的单位阵相乘等，这些都有特定的求解公式，在此不做深入，只讨论最基本的乘法。

在C++中，实现该算式需要一个三重循环将目标阵、原阵依次遍历，前两层循环遍历C矩阵的每一个元素，最后一层循环实现上述的求和公式，此算法虽然经典，但时间复杂度达到了O(n3)，计算时间太长，不适用于实际运算。

循环代码如下文：

for (i = 0; i < m; i++)

for (j = 0; j < n; j++)

for (s = 0; s < n; s++)

c[i][j] += a[i][s] \* b[s][j];

二、分治法求解

分治要求选取方阵作为研究对象，其中n为2的幂级数，以A=（aij）n\*n和B=(bij)n\*n为例,分治法求解矩阵利用了矩阵的分块性质，将较大矩阵的乘法转换为较小的矩阵乘法，

AB= A11 A12 B11 B12 = A11B11+A12B21  A11B12+A12B22

A21 A22 B21 B22  A21B11+A22B21 A21B12+A22B22

1.1基本分治法

最简单的分治理方法就是按照这个公式对矩阵进行分割之后合并，先判断矩阵的维数是否为二阶，如果是则可以直接进行求解，如果不是就将矩阵分割成4个小的矩阵之后进行递归求解，并将每次求解结果合并成为一个新的矩阵再进行运算，最后返回一个最终的运算结果矩阵C。

伪代码：

multip(Matrix A,Matrix B){

if(row==2)

C=Result(Martix A,Matrix B), return C;

else{

A11=Divide(A,11); … B22=Divide(B,22);

C11=Add(multip(A11,B11),multip(A12,B21)); …

return combine(C11,C12,C21,C22);

}

}

1.2算法分析

该算法采用了一定的分治思想，在每次递归中都会产生四个子阵，之后进行四次乘法操作，由于矩阵没分为了四个子块，递归的深度大概是log2(n),故总的计算的次数大概是4log2(n)\*4=4\*n2。T(n)=4\*T(n/2)+O(n2)(a=4,b=2),运用公式解得算法将矩阵计算的时间复杂度降到了O(n2)量级，在一定规模下比传统的矩阵乘法更加高效。

该算法每次递归的分割都使用了四个子阵的空间，递归栈月占用O(log(n))的空间。算法使用了临时矩阵来存储子块的乘积，临时矩阵占用的空间复杂为O(n2),最终为结果分配了一定的存储空间，与输入矩阵大小相关，占用较小，总的空间复杂度约为O(log(n))+ O(n2)=O(n2)。

2.1经典Strassen算法

上述算法实现的时间性能与Strassen相当，但是需要太多的内控以存储递归栈和临时矩阵。该算法在求解中使用了8次乘法，对于规模较小的矩阵，递归次数较少，适合使用该方法，但当矩阵规模变大，递归次数增加，算法求解劣势明显。

Strassen算法采用了一个数学技巧，将每次递归中的8次乘法变为了7次乘法，将传统乘法的效率提高了1/8。

伪代码：

multip(A,B,C){

T1=A11+A22;t2=B11+B22;multip(T1,T2,M1); T3=A21+A22;multip(T3,B11,M2); T4=B12-B22;multip(A11,T4,M3); T5=B21-B11;multip(A22,T5,M4); T6=A11+A12;multip(T6,B22,M5); T7=A21-A11;T8=B11+B12;multip(T7,T8,M6);

T9=A12-A22;T10=B21+B22;multip(T9,T10,M7);

C11=M1+M4-M5+M7; C12=M3+M5; C21=M2+M4; C22=M1-M2+M3+M6;

}

2.2分析Strassen算法

基本的Strassen算法在每次递归中仍然将矩阵分为四个子阵，递归深度仍然是log2(n)，在进行了10次基本加减法之后，需要进行7次乘法运算，故总的计算次数约为7\*4log2(n)。而T(n)=7\*T(n/2)+O(n2)(a=7,b=2),运用公式可解得算法的时间复杂度约为O(nlog2(7))=O(n2.81)。

算法每次递归是都会产生四个子阵，占用相应的存储空间，因此递归栈的空间复杂度是O(log(n))。每一层递归中又会产生7个新的子阵，临时矩阵占用的总存储空间约为7\*n2/4，复杂度为O(n2)。最后结果矩阵占用的空间与输入的矩阵规模相关，相比于临时矩阵和递归栈较小。故总的空间复杂度为O(log(n))+ O(n2)= O(n2)。

三、总结

上述两种算法都通过分治法将传统矩阵的运算简化。第一种算法作为简单的分治法，算法更容易理解和实现，不需要对算法本身进行太大的调整，在递归过程中生成的多个子问题都可求解。但与Strassen算法相比，这种简单的分块方法需要更多的乘法操作，使得算法的时间效率略逊一筹。同时算法的内存占用较大，不利于普及于某些机器。这特定的应用场景下，这两种算法应该都有各自的优势和劣势，具体的求解方法要基于实际给出的矩阵进行选择。

参考文献：

[1]https://blog.csdn.net/qq\_42642142/article/details/107546325

[2]龙腾芳.矩阵乘法的两个算法分析[J].韶关学院学报(自然科学版),2001(09):6-9.

[3]https://www.cnblogs.com/wonderKK/archive/2011/11/07/2240262.html