Implementing matrix operations for a subclass of block matrices

Siddharth Bhat
<siddu.druid@gmail.com>
github.com/bollu

August 16, 2018

1 Introduction

We study the problem of implementing specialized operations for matrix multiplication and matrix inversion on a subclass of matrices.

These matrices are parametrized by two variables, henceforth denoted by S, and D. Such matrices live in the space of $\mathbb{R}^{BD \times BD}$. They consist of non-overlapping diagonal blocks of size $D \times D$, arranged in a grid of S^2 such blocks. An example of such a matrix is below:

$$\begin{bmatrix} D_{11} & D_{12} & \dots & D_{1S} \\ \dots & \dots & \dots & \dots \\ D_{S1} & D_{S2} & \dots & D_{SS} \end{bmatrix}$$

This parametrization will be dubbed the **character** (S, D) of the matrix. S for ”size”, and D for ”diagonal”.

If we set $D = 1, S = *$, we regain the usual matrcies that we use, of dimension $S \times S$. Simiarly, if we set $D = *, S = 1$, we regain diagonal matrices of dimension $D \times D$. These two extreme cases will be used repeatedly to sanity check our formulas, and make for useful examples to look back at.

After all, to quote Paul Halmos, - “…the source of all great mathematics is the special case, the concrete example. It is frequent in mathematics that every instance of a concept of seemingly great generality is in essence the same as a small and concrete special case.”

2 Matrix representation

Note that we only need to store the diagonal elements of all the blocks of any given matrix. Therefore, in our representation, we choose to save the diagonal elements per-block, giving us a representation of the matrix as a collection of $S \times S \times D$ numbers, which are indexed as: $\langle$`row block index, column block index, diagonal index`$\rangle = \langle$`r, c, d`$\rangle$

The amount of space required per block will be $\mathcal{O}(D)$. Since the total number of blocks is S^2, the total space complexity of this matrix will be $\mathcal{O}(S^2 D)$. This is better than storing an entire $S \times D$-dimensional matrix which will work out to $\mathcal{O}((S \times D)^2) = \mathcal{O}(S^2 D^2)$. We save space by a factor of D.

3 Matrix multiplication

3.1 Matrcies that have the same (S, D) character

Here, we consider multiplying two matrices which have identical D, S. This will be used as a stepping stone in the theory of general matrix multiplication. This is also more faithful to how I understood the problem, so I choose to perform the exposition along a similar path.

3.1.1 A quick aside: Block multiplication

When we are multiplying matrices, we can multiply them as blocks - that is, this formula holds:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

Where A_{ij}, B_{ij} are blocks of the original matrix. This also generalises to any number of blocks we wish to choose. The rough proof is that to compute an element of the output matrix, c_{ij}, we compute it as $c_{ij} = \sum_k a_{ik}b_{kj}$. Thinking computationally, the computation of matrix C only ever reads from A and B. Thus, we can perform the computation of elements of C in any order we want, and computing per-block is a perfectly valid choice.

So now, if we try to analyze how to compute C_{11}, we realize the that the computation of elements of C_{11} can be factored as shown above. A similar argument holds for all blocks of C.

3.1.2 Exploiting block multiplication

Now that we know that block multiplication is possible, an immediate solution arises — we already have blocks of size $D \times D$, which are extremely easy to multiply (since they are diagonal matrices). Hence, we block the two input matrices A and B into their diagonal blocks (they are both of the same character (S, D)).

This leads us to the algorithm (all code will be in python-like pseudocode)

```
def matmulDiagEqCharacter(m1, m2):
    (S1, D1) = matdiagdim(m1)
    (S2, D2) = matdiagdim(m2)

    # sanity check inputs
    assert S1 == S2 and D1 == D2

    # dimensions of the output matrix
    SNEW = S1
    DNEW = D1
    out = zeromat(S, D)

    for i in range(SNEW):
        for j in range(SNEW):
            for k in range (SNEW):
                for d in range(DNEW):
                    out[i][j][d] += m1[i][k][d] * m2[k][j][d]

    return out
```

Clearly (from the `for`-loops in the code), the time complexity of this algorithm works out to $\mathcal{O}(S^3 D)$, which in comparison to the naive $\mathcal{O}((S \times D)^3) = \mathcal{O}(S^3 D^3)$ saves us a factor of around D^2.

3.2 Matrices with different (S, D) character

Let us consider trying to multiply matrices of different (S, D) characters. Let A have character $(S1, D1)$, and B be $(S2, D2)$. Since these are square matrices, we can arrive at the condition that $S1 \times D1 = S2 \times D2$. However, once we have analyzed this, we need some insight: Will the resultant always be a (S', D') matrix? If so, why? And how do we compute the resultant (S', D') values?

Let's try some examples. We already know that if $S1 = S2, D1 = D2$, then $S1' = S1 = S2, D' = D1 = D2$.

The other example we can take is that of $(S1, D1) = (1, N)$ and $(S2, D2) = (N, 1)$. In this case, A is a diagonal matrix, and B is a "normal" matrix. We know the result is another normal matrix, so $(S' = N, D' = 1)$.

We tentatively conjecture that something along the lines of `gcd` or `lcm` should be involved in this process, since we need to find a new blocking size D', which would need to properly divide $D1$, $D2$ intuitively for our arguments to carry over.

I ran some experiments at this point and experimentally verified that $D' = gcd(D1, D2)$ is the correct relation. This fits with our previous examples as well. Once I had this, I began looking for an explanation. Note that we can block the original matrix A into blocks of size D', since

D' divides $D1$. Simiarly, we can block B as well into blocks of size D', since D' also divides $D2$.

This is a legal blocking, since the original blocks $D1$ get subdivided into blocks D' which **still only have elements on the diagonal**. Similarly, the "empty" blocks that are created from $D1$