Wavelet Multiplication

Daniel Beatty

June 26, 2003

1 Wavelet Matrix Multiplication

One of the key points for wavelet matrix multiplication is the proof that $W(A) \times W(B) = W(A \times B)$ If this is the case, then it is obvious that $W(A) \times W(B) = W(C) = W(A \times B = C)$. So far the proof is still weak. The reason is that an example proof is useful for proving something to not be the case, rather than being the case. However, a simple example does show some intuitive steps that would be necessary for a proof.

1.1 A 2×2 example

1.1.1 Conventional Multiplication

Conventional multiplication is spelled out as

$$c_{i,j} = \sum_k a_{i,k} b_{k,j}$$

For a 2×2 matrix, there is the following:

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

1.1.2 Wavelet Transform of two $2\times$ 2 matrices

For a wavelet transform on both matrix A and B, the results are:

$$W(A) = \frac{1}{2} \begin{pmatrix} a_{1,1} + a_{2,1} + a_{1,2} + a_{2,2} & a_{1,1} + a_{2,1} - a_{1,2} - a_{2,2} \\ a_{1,1} - a_{2,1} + a_{1,2} - a_{2,2} & a_{1,1} - a_{2,1} - a_{1,2} + a_{2,2} \end{pmatrix}$$

$$W(B) = \frac{1}{2}\begin{pmatrix} b_{1,1}+b_{2,1}+b_{1,2}+b_{2,2} & b_{1,1}+b_{2,1}-b_{1,2}-b_{2,2} \\ b_{1,1}-b_{2,1}+b_{1,2}-b_{2,2} & b_{1,1}-b_{2,1}-b_{1,2}+b_{2,2} \end{pmatrix}$$

1.1.3 Product of A and B in wavelet space

The conventional product of A and B can be transformed into wavelet space. The results of this matrix transform is as follows:

$$W(A\times B) = W\begin{pmatrix} a_{1,1}b_{1,1}+a_{1,2}b_{2,1} & a_{1,1}b_{1,2}+a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1}+a_{2,2}b_{2,1} & a_{2,1}b_{1,2}+a_{2,2}b_{2,2} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} (a_{1,1}b_{1,1}+a_{1,2}b_{2,1}+a_{1,1}b_{1,2}+a_{1,2}b_{2,2})+(a_{2,1}b_1 \\ (a_{1,1}b_{1,1}+a_{1,2}b_{2,1}+a_{1,1}b_{1,2}+a_{1,2}b_{2,2})-(a_{2,1}b_1 \end{pmatrix}$$

$$W(A\times B) == \frac{1}{2}\begin{pmatrix} (a_{1,1}b_{1,1}+a_{1,2}b_{2,1}+a_{1,1}b_{1,2}+a_{1,2}b_{2,2}+a_{2,1}b_{1,1}+a_{2,2}b_{2,1}+a_{2,1}b_{1,2}+a_{2,2}b_{2,2}) & (a_{1,1}b_{1,1}+a_{1,2}b \\ (a_{1,1}b_{1,1}+a_{1,2}b_{2,1}+a_{1,1}b_{1,2}+a_{1,2}b_{2,2}-a_{2,1}b_{1,1}-a_{2,2}b_{2,1}-a_{2,1}b_{1,2}-a_{2,2}b_{2,2}) & (a_{1,1}b_{1,1}+a_{1,2}b \end{pmatrix}$$

1.1.4 What is $W(A) \times W(B)$

Straight forward multiplication of $W(A) \times W(B)$ works out as follows:

$$W(A)\times W(B) = \frac{1}{2}\begin{pmatrix} a_{1,1}+a_{2,1}+a_{1,2}+a_{2,2} & a_{1,1}+a_{2,1}-a_{1,2}-a_{2,2} \\ a_{1,1}-a_{2,1}+a_{1,2}-a_{2,2} & a_{1,1}-a_{2,1}-a_{1,2}+a_{2,2} \end{pmatrix}\times\frac{1}{2}\begin{pmatrix} b_{1,1}+b_{2,1}+b_{1,2}+b_{2,2} & b_{1,1}+b_{2,} \\ b_{1,1}-b_{2,1}+b_{1,2}-b_{2,2} & b_{1,1}-b_{2,} \end{pmatrix}$$

Of course this is better simplified.

$$\frac{1}{2}\begin{pmatrix} (a_{1,1}b_{1,1}+a_{2,1}b_{1,1}+a_{1,2}b_{2,1}+a_{2,2}b_{2,1}+a_{1,1}b_{1,2}a_{2,1}b_{1,2}+a_{1,2}b_{2,2}+a_{2,2}b_{2,2}) & (a_{1,1}b_{1,1}+a_{2,1}b_{1,1}+a_{1,2}b_{2,1}+ \\ (a_{1,1}b_{1,1}-a_{2,1}b_{1,1}+a_{1,2}b_{2,1}-a_{2,2}b_{2,1}+a_{1,1}b_{1,2}-a_{2,1}b_{1,2}+a_{1,2}b_{2,2}-a_{2,2}b_{2,2}) & (a_{1,1}b_{1,1}-a_{2,1}b_{1,1}+a_{1,2}b_{2,1}- \end{pmatrix}$$

Compared with $W(C)$ computed the conventional way:

$$W(C) = \frac{1}{2}\begin{pmatrix} (a_{1,1}b_{1,1}+a_{1,2}b_{2,1}+a_{1,1}b_{1,2}+a_{1,2}b_{2,2}+a_{2,1}b_{1,1}+a_{2,2}b_{2,1}+a_{2,1}b_{1,2}+a_{2,2}b_{2,2}) & (a_{1,1}b_{1,1}+a_{1,2}b_{2,1}- \\ (a_{1,1}b_{1,1}+a_{1,2}b_{2,1}+a_{1,1}b_{1,2}+a_{1,2}b_{2,2}-a_{2,1}b_{1,1}-a_{2,2}b_{2,1}-a_{2,1}b_{1,2}-a_{2,2}b_{2,2}) & (a_{1,1}b_{1,1}+a_{1,2}b_{2,1}- \end{pmatrix}$$

Notice that $W(A) \times W(B) = W(A \times B)$, in the case of 2×2 matrices.

1.2 Proof of Wavelet Matrix Multiplication

A reminder from Dr. Sinzinger, wavelets and their inverses are linear operators. As he correctly reminded me any linear operator has the following properties:

1. $L(AB) = L(A)\dot{L}(B)$
2. ψ^{-1} is a linear operator
3. $\psi^{-1}(\psi(A)\dot{\psi}(B)) = \psi^{-1}(\psi(A))\dot{\psi}^{-1}(\psi(B))$
4. $\psi^{-1}(\psi(A)) = A$
5. $\psi^{-1}(\psi(B)) = B$
6. Therefore: $AB = \psi^{-1}(\psi(A)\dot{\psi}(B))$

Thus this is sufficient proof that wavelet matrix multiplication is sound.

2 Chain Multiplication Structure

Recall the chain structure. This structure can be referenced from most introductory algorithm books. The big idea is to be able to locate specific items quickly by some key which reduces the search to a family of keys that close together by a hashing scheme. In the case of matrix multiplication, no hash key is needed. The row or column identifier is sufficient for this. Of course a description of this scheme is in order.

Each chain is organized as a one-dimensional array. Each array has an array list, called a link, which contains the actual data, and array list count showing how many items are in each array. A next and previous value in this case would be superfluous since columns and rows are arranges in lexicographical order. The array list contains the following items:

- key
- item value (row value/ column value)
- previous key
- next key

In the case of a column chain, the keys are row identifiers. In the case of a row chain, the keys are column identifier. This structure is similar to a matrix and can be represented by a matrix. The essential thing for the chain is the arrangement of the array lists. Zero values are not allowed, and thus order and index keys must be maintained.

What has this to do with sparse matrix multiplication? The left matrix is transformed into a row chain and the right matrix is transformed into a column chain. Each value of the result matrix is simply a multiplication of the row vectors in the left matrix by the column vectors in the right matrix. With the chain structure, it simply row links times the column links. Only elements with matching keys are allowed to be multiplied together. The rest are assumed to be zero, and thus no action is taken. The sum of the multiplies is the result. The multiplication procedure is as follows:

1. Chain Multiply
   (a) Arguments: left matrix chain (A) and right matrix chain (B)
   (b) Results: Result matrix

   - $\forall i \in R.row$
     - $\forall j \in R.col$
       * $R_{i,j} = CM(A[i], B[j])$
       * —- Note that CM is the chain multiply procedure.

2. Chain Multiply Element
   (a) Arguments: row link (r) and column link (c)
   (b) Output: Double result: total

   - rlimit = r.size;
   - climit = c.size;
   - k = l = 0;
   - jlow = 0;
   - total = 0.0
   - BnotExhausted = true
   - while both (k ¡ rlimit) and (BnotExhausted)
     - $\forall l \in [jlow, climit)$ if ($A_i^c.getkey(k) \equiv B_j^c getkey(l)$) then lmatch = l
     - if ($l \equiv climit$) BnotExhausted = false
     - otherwise
       * temp += $A_i^c[k] * B_i^c[lmatch]$
       * jlow = lmatch

For the Matrix Chain Multiply the complexity is $O(N^2)$. For the chain multiply, the complexity is $O(M)$ where M is the larger length of the two links. Thus total complexity is $O(N^2M)$ since M's largest size is N. This a general and simple algorithm for sparse matrix multiplication of matrix chains. Of course the chore of loading the matrix chains is $O(N^2)$ per matrix. Thus total cost is on the order of $O(3N^2M)$. So long as M is significantly less than N (on the order of a $\frac{1}{3}$), then the wavelet matrix multiplication has a reasonable advantage.

3 Practical Implementation

There are few questions about the practical implementation that must be answered:

1. Are the matrices square matrices, and same size?
2. If not, are the dimensions suitable for multiplication?
3. If so, what is the maximum resolution for each matrix? The lesser maximum is the limit for both.
4. Is the wavelet transform performed on each matrix the same?

A yes answer to the first matrix, allows for an relatively easy transform, and matrix multiply. If the matrices are not square, then practical problems exist. If the wavelet transform is the same on each, then the linear proof is sound. However, if they are not then the proof is bogus and the multiplication is too.

The general wavelet based matrix multiply is as follows:

1. Arguments:
   - A : a $m \times p$ matrix
   - B: a $p \times n$ matrix
2. Results: C : a $m \times n$ matrix
3. Procedure:
   - $A \xrightarrow{\psi} \alpha$
   - $B \xrightarrow{\psi} \beta$
   - $\alpha \overset{ChainRow}{\rightarrow} \alpha^c$
   - $\beta \overset{ChainColumn}{\rightarrow} \beta^c$
   - Chain Multiply $(\alpha^c, \beta^c) \rightarrow C$

3.1 Chain Row and Chain Column Setup

Setting up the the chain-link structure for row or column orientation is relatively simple. Each one requires proper addressing of the hooks. The procedures are as follows:

Row Chain-Link Setup

- $\forall i \in \alpha.rows$

- $\forall j \in \alpha.columns$
  - if ($\alpha[i][j] \approx 0$) $a^c.hook_i.addlink(j, \alpha_{i,j})$

Column Chain Link Setup

- $\forall i \in \alpha.rows$
  - $\forall j \in \alpha.columns$
    - if ($\alpha[i][j] \approx 0$) $a^c.hook_i.addlink(j, \alpha_{i,j})$

One of the big questions is how to optimize wavelet packets for multiplication applications. A straight multi-resolution wavelet on a square matrix is seemingly trivial. In the square matrix, straight multi-resolution case the number of resolutions is dictate by the size of the matrices being multiplied. For wavelet packets on square matrices, it is also a size issue. One can apply wavelet packets to maximum extent that the matrix size allows. However, some of the packet levels may be unnecessary since the amount energy shuffling is small in comparison to the number of operations. In the case of non-square matrices, best fits must be applied to ensure that the same wavelet transform

The Chain-Link Structure

The chain link structure is derived from 2 other classes (hooks and links). Each one of these classes are relatively simple and can be implemented with either arrays or pointers. Due to the anticipated size, the array mechanism is chosen for speed and efficiency.

Class Chain Link members

- length
- hooks

Class Hook members:

- length
- l2norm
- links

Class Link members:

- id
- value

Recursive Wavelet Packets The algorithm for optimal wavelet packets is easiest to describe by recursive means. The stop condition either when the maximum resolution has been reached, or energy shuffle metric is satisfied.

procedure wavepack (matrix A, integer res) return μ

- if ($res \equiv 0$) stop
- if ($A.rowmodulo2 \equiv 0$ and $A.colmodulo2 \equiv 0$) stop
- if (optimum) stop
- otherwise
  - $A \stackrel{\psi}{\rightarrow} \mu$
  - $\mu \stackrel{topleft}{\rightarrow} \alpha$
  - $\mu \stackrel{lowleft}{\rightarrow} \beta$
  - $\mu \stackrel{topright}{\rightarrow} \gamma$
  - $\mu \stackrel{lowright}{\rightarrow} \delta$
  - wavepack (α , res -1)
  - wavepack (β , res -1)
  - wavepack (γ , res -1)
  - wavepack (δ , res -1)
  - insert ($\alpha \stackrel{topleft}{\rightarrow} \mu$)
  - insert ($\gamma \stackrel{topright}{\rightarrow} \mu$)
  - insert ($\beta \stackrel{lowleft}{\rightarrow} \mu$)
  - insert ($\delta \stackrel{lowright}{\rightarrow} \mu$)
  - stop

The determining of the optimum condition is matter that is not well defined.