A NORMAL FORM FOR MATRIX MULTIPLICATION SCHEMES

Manuel Kauers and Jakob Moosbauer*

27.10.2022

JOHANNES KEPLER
UNIVERSITY LINZ

A Normal Form for Matrix Multiplicatoin Schemes

- Matrix Multiplication Schemes
- Equivalence Classes
- Normal Form

Matrix Multiplication

- The standard algorithm for multiplying two $n \times n$ matrices uses $\mathrm{O}(n^3)$ operations in the base ring.
- Strassen's algorithm can do the job in $\mathrm{O}(n^{2.81})$ operations.
- The best asymptotic bound is $\mathrm{O}(n^{2.372})$ [Alman & Williams, 2020].
- Recently an algorithm was found that multiplies matrices over $\mathbb{Z}_2$ in $\mathrm{O}(n^{2.78})$ operations [Fawzi et al., 2022].
- Almost all algorithms which are efficient in practice are based on concrete multiplication schemes for small matrices.
- For 2×2 matrices, we know that 7 multiplications are necessary and sufficient and Strassen's algorithm is - in some sense - the only such algorithm.
- Already in the 3×3 case there is known much less.

Matrix Multiplication Schemes

$$
\begin{aligned}
m_1 &= a_{1,1}b_{1,1} \\
m_2 &= a_{1,2}b_{2,1} \\
m_3 &= a_{1,1}b_{1,2} \\
m_4 &= a_{1,2}b_{2,2} \\
m_5 &= a_{2,1}b_{1,1} \\
m_6 &= a_{2,2}b_{2,1} \\
m_7 &= a_{2,1}b_{1,2} \\
m_8 &= a_{2,2}b_{2,2} \\
c_{1,1} &= m_1 + m_2 \\
c_{1,2} &= m_3 + m_4 \\
c_{2,1} &= m_5 + m_6 \\
c_{2,2} &= m_7 + m_8
\end{aligned}
$$

$$
\begin{aligned}
m_1 &= (a_{1,1} + a_{2,2})(b_{1,1} + b_{2,2}) \\
m_2 &= (a_{1,1} + a_{1,2})(b_{2,2}) \\
m_3 &= (a_{2,1} + a_{2,2})(b_{1,1}) \\
m_4 &= (a_{1,1})(b_{1,2} - b_{2,2}) \\
m_5 &= (a_{2,2})(b_{2,1} - b_{1,1}) \\
m_6 &= (a_{2,1} - a_{1,1})(b_{1,1} + b_{1,2}) \\
m_7 &= (a_{1,2} - a_{2,2})(b_{2,1} + b_{2,2}) \\
\\
c_{1,1} &= m_1 - m_2 + m_5 + m_7 \\
c_{1,2} &= m_2 + m_4 \\
c_{2,1} &= m_3 + m_5 \\
c_{2,2} &= m_1 - m_3 + m_4 + m_6
\end{aligned}
$$

Matrix Multiplication Schemes

$$
\begin{aligned}
m_1 &= (\alpha^{(1)}_{1,1} a_{1,1} + \alpha^{(1)}_{1,2} a_{1,2} + \ldots)(\beta^{(1)}_{1,1} b_{1,1} + \beta^{(1)}_{1,2} b_{1,2} + \ldots) \\
&\vdots \\
m_7 &= (\alpha^{(7)}_{1,1} a_{1,1} + \alpha^{(7)}_{1,2} a_{1,2} + \ldots)(\beta^{(7)}_{1,1} b_{1,1} + \beta^{(7)}_{1,2} b_{1,2} + \ldots) \\
c_{1,1} &= (\gamma^{(1)}_{1,1} m_1 + \ldots + \gamma^{(7)}_{1,1} m_7) \\
&\vdots \\
c_{2,2} &= (\gamma^{(1)}_{2,2} m_1 + \ldots + \gamma^{(7)}_{2,2} m_7)
\end{aligned}
$$

The coefficients need to be such that

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}.$$

The Brent Equations

$$\sum_{l=1}^{r} \alpha^{(l)}_{i_1,i_2} \beta^{(l)}_{j_1,j_2} \gamma^{(l)}_{k_1,k_2} = \delta_{i_2,j_1} \delta_{i_1,k_1} \delta_{j_2,k_2}$$

$$i_1, i_2, j_1, j_2, k_1, k_2 \in \{1, \ldots, n\}$$

System with $3rn^2$ variables and n^6 cubic equations.
Observe:

- Permuting the upper indices only changes the order of the multiplications.
- This transformation leaves the equations unchanged.
- We can permute α, β and γ if we swap the indices accordingly.

Tensor Representation

To view a multiplication scheme as an algebraic object we write the coefficients in a tensor:

$$\sum_{l=1}^{7} \begin{pmatrix} \alpha_{1,1}^{(l)} & \alpha_{1,2}^{(l)} \\ \alpha_{2,1}^{(l)} & \alpha_{2,2}^{(l)} \end{pmatrix} \otimes \begin{pmatrix} \beta_{1,1}^{(l)} & \beta_{1,2}^{(l)} \\ \beta_{2,1}^{(l)} & \beta_{2,2}^{(l)} \end{pmatrix} \otimes \begin{pmatrix} \gamma_{1,1}^{(l)} & \gamma_{1,2}^{(l)} \\ \gamma_{2,1}^{(l)} & \gamma_{2,2}^{(l)} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}$$

Symmetries

If we exchange α and β in all the equations

$$\sum_{l=1}^{r} \alpha^{(l)}_{i_1,i_2} \beta^{(l)}_{j_1,j_2} \gamma^{(l)}_{k_1,k_2} = \delta_{i_2,j_1} \delta_{i_1,k_1} \delta_{j_2,k_2},$$

we have to swap all indices to preserve the same system of equations.

This corresponds to the fact that $(AB)^T = B^T A^T$.

Symmetries

We can find another symmetry corresponding to $AUU^{-1}B = AB$ for an invertible matrix U. It turns out that if

$$\sum_{l=1}^{r} A^{(l)} \otimes B^{(l)} \otimes C^{(l)}$$

is a matrix multiplication scheme and U, V, W are invertible $n \times n$ matrices, then

$$\sum_{l=1}^{r} UA^{(l)}V^{-1} \otimes VB^{(l)}W^{-1} \otimes WC^{(l)}U^{-1}$$

is as well.

Symmetry Group

De Groote proved that Strassen's algorithm is unique up to this symmetry group.

- (Reordering rows $\to S_r$)
- Permuting α, β and $\gamma \to S_3$
- Applying a triple of matrices $(U, V, W) \in \mathrm{GL}(K, n)^3$

We restrict ourselves to the case that K is a finite field. For example for $\mathbb{Z}_2$ we have:

$$|\mathrm{GL}(\mathbb{Z}_2, 2)^3| = 216$$
$$|\mathrm{GL}(\mathbb{Z}_2, 3)^3| = 4.741.632$$
$$|\mathrm{GL}(\mathbb{Z}_2, 4)^3| = 8.193.540.096.000$$

Recognizing Equivalences

Definition

We call two multiplication schemes $(A^{(l)}, B^{(l)}, C^{(l)})_{l=1}^{r}$ and $(A'^{(l)}, B'^{(l)}, C'^{(l)})_{l=1}^{r}$ equivalent if there is an element $g \in S_r \times S_3 \times \mathrm{GL}(K, n)$, such that

$$g * (A^{(l)}, B^{(l)}, C^{(l)})_{l=1}^{r} = (A'^{(l)}, B'^{(l)}, C'^{(l)})_{l=1}^{r}.$$

There is an algorithm to check wether two schemes are equivalent. To find out whether a given scheme is equivalent to any of a set of known schemes we would need to go over them one by one. It is more useful to have a canonic representative for every equivalence class.

Rank Patterns

The ranks of the matrices $A^{(l)}, B^{(l)}, C^{(l)}$ are invariant under the action of $\mathrm{GL}(\mathbb{Z}_2, n)^3$.

Definition

Given a multiplication scheme $(A^{(l)}, B^{(l)}, C^{(l)})_{l=1}^{r}$ we call the table $(\mathrm{rank}(A^{(l)}), \mathrm{rank}(B^{(l)}), \mathrm{rank}(C^{(l)}))_{l=1}^{23}$ the rank pattern of the scheme.

Since the rank is invariant under transposition the full symmetry group only permutes the rows and columns of the rank pattern. Thus, a sorted rank pattern is invariant under the action of the symmetry group.

Rank Pattern

3	1	1
2	2	2
2	2	2
2	2	2
2	2	1
2	1	1
1	3	1
1	2	2
1	1	1
1	1	1
1	1	1

Normal Forms

A very primitive normal form computation would be to go over the complete orbit and determine the lexicographically smallest element.

We can do better if we first determine the maximal rank pattern. We can then restrict to permutations that leave this rank pattern invariant and only have to go through $\mathrm{GL}(K, n)^3$.

So we define the Normal Form to be the lexicographically smallest element of an equivalence class that has the maximal rank pattern.

Normal Form Computation

We consider the order of the columns in the scheme to be fixed.

1. Initalize $head$ to the empty list.
2. Initalize $tails$ to the set containing the given scheme.
3. Initialize $stab$ to $\mathrm{GL}(K, n)^3$.
4. Determine the smallest possible next row in the normal form from all rows in every element of $tails$ under the action of $stab$.
5. Append this row to $head$ and adjust $stab$ and $tails$.
6. Set $stab$ to the stabilizer of $head$.
7. Repeat from 4. until done.

Normal Form Computation

- The naive way to compute a normal form takes time $\mathrm{O}(r \cdot \left|\mathrm{GL}(K,n)^3\right|)$.
- The presented algorithm is more efficient becuase the stabilizers become very small very quickly.
- In practice in almost all cases we can expect that after $\mathrm{O}(1)$ steps the stabilizer has size $\mathrm{O}(1)$.
- Under this assumpation the complexity is reduced to $\mathrm{O}(r + \left|\mathrm{GL}(K,n)^3\right|)$.
- The problem remains that for the first row we have to go over the full group $\mathrm{GL}(K,n)^3$.

Minimizing the first row

We order matrices using colexicographic order by columns. A row (A, B, C) that is minimal under the action of $\mathrm{GL}(K, n)^3$ has the following properties:

1. A has the form
$$\begin{pmatrix} 0 & 0 \\ I_r & 0 \end{pmatrix}$$
where $r = \operatorname{rank} A$.
2. B is in column echelon form.
3. If $\operatorname{rank} A = n$, then $A = I_n$ and B has the form
$$\begin{pmatrix} 0 & 0 \\ I_r & 0 \end{pmatrix}$$
where $r = \operatorname{rank} B$.

Minimizing the first row

Given a row (A, B, C) we can determine the smallest equivalent row $(A_1, B_1, C_1) = (UAV^{-1}, VBW^{-1}, WCU^{-1})$ and it's stabilizer the following way:

- If $\operatorname{rank}(A) = n$, then $A_1 = I_n$.
  - In this case we can choose $U = (AV^{-1})^{-1} = VA^{-1}$ and start by minimizing B.
  - If B has full rank as well, we set $V = WB^{-1}$ and determine W such that WCW^{-1} is minimal.
- Otherwise, we can compute the stabilizer of A_1 by solving the linear system $SA_1T^{-1} = A_1$ and discarding all singular solutions.
  - For every such T in the stabilizer of A_1 we compute all invertible matrices R that minimize TBR^{-1}.
  - From all triples U, V, W select those that minimize C as well.

Final Analysis

- In the worst case we still have to iterate over a stabilizer of size $\Omega(\mathrm{GL}(K,n)^3)$.
- However, in our experiments all stabilizers turned out to be much smaller.
- Under the assumption that none of the stabilizers exceeds $\mathrm{O}(\left|\mathrm{GL}(K,n)^2\right|)$, the complexity of the normal form computation is $\mathrm{O}(r + \left|\mathrm{GL}(K,n)^2\right|)$
- In our experiments for 3×3 matrix multiplication schemes over $\mathbb{Z}_2$ we observed that computing normal forms is more efficient than direct comparison if we have more than 200 schemes.