Finding Faster Matrix Multiplication Algorithms with Reinforcement Learning

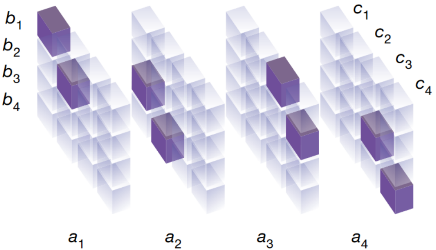
Fawzi et al.

Motivation

Matrix multiplication: multiplying two $n \times n$ matrices takes $O(n^{\omega})$

- $\omega < \log_2 7 \approx 2.808$ [Strassen69]

$$\begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix}$$



$$m_1 = (a_1 + a_4)(b_1 + b_4)$$
$$m_2 = (a_3 + a_4)\,b_1$$
$$m_3 = a_1\,(b_2 - b_4)$$
$$m_4 = a_4\,(b_3 - b_1)$$
$$m_5 = (a_1 + a_2)\,b_4$$
$$m_6 = (a_3 - a_1)(b_1 + b_2)$$
$$m_7 = (a_2 - a_4)(b_3 + b_4)$$
$$c_1 = m_1 + m_4 - m_5 + m_7$$
$$c_2 = m_3 + m_5$$
$$c_3 = m_2 + m_4$$
$$c_4 = m_1 - m_2 + m_3 + m_6$$

Motivation

Latest asymptotic results: $\omega < 2.37287$ [Le Gall14], $\omega < 2.37286$ [AV20]

- Small improvements
- Highly technical
- Impractical

Machine learning

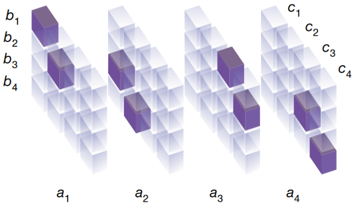
- New perspective
- Computer assistance
- Flexible objective

MM Algorithms as tensor decompositions

- MM can be represented by tensors $T_n \in \{0,1\}^{n^2} \otimes \{0,1\}^{n^2} \otimes \{0,1\}^{n^2}$
- A (Strassen-like) algorithm is a rank-one decomposition

$$T_n = \sum_t u^{(t)} \otimes v^{(t)} \otimes w^{(t)}, \text{ where } u, v, w \in \mathbb{R}^{n^2}$$

$$\begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix}$$



$$m_1 = (a_1 + a_4)(b_1 + b_4)$$
$$m_2 = (a_3 + a_4)\, b_1$$
$$m_3 = a_1 (b_2 - b_4)$$
$$m_4 = a_4 (b_3 - b_1)$$
$$m_5 = (a_1 + a_2)\, b_4$$
$$m_6 = (a_3 - a_1)(b_1 + b_2)$$
$$m_7 = (a_2 - a_4)(b_3 + b_4)$$
$$c_1 = m_1 + m_4 - m_5 + m_7$$
$$c_2 = m_3 + m_5$$
$$c_3 = m_2 + m_4$$
$$c_4 = m_1 - m_2 + m_3 + m_6$$

$$\mathbf{U} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}$$

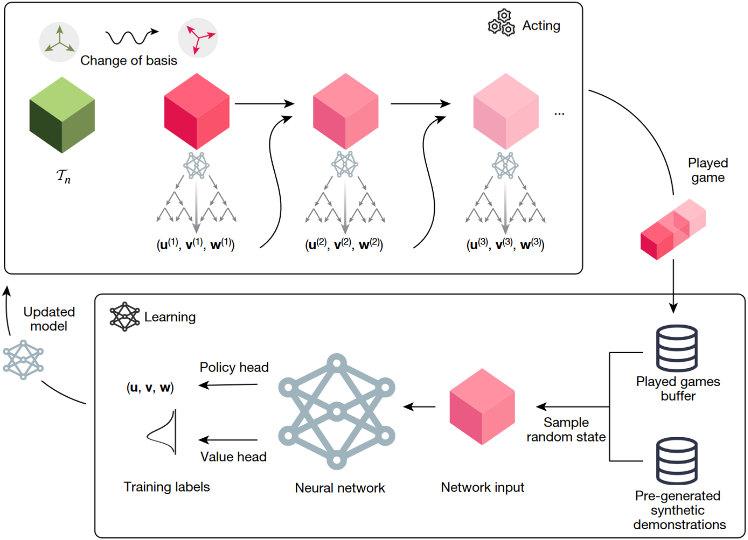
$$\mathbf{V} = \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Formulating MM Algorithm as Reinforcement Learning

- Similar to AlphaGo, etc.
- States: S_t with $S_0 = T_n$
- Actions: $u^{(t)} \otimes v^{(t)} \otimes w^{(t)}$, where $u, v, w \in F^{n^2}$, F is e.g. $\{0, \pm 1\}$
- Update: $S_t \leftarrow S_{t-1} - u^{(t)} \otimes v^{(t)} \otimes w^{(t)}$
- Termination: $t = R$ or $S_t = 0$
- Evaluation: $-R - \mathrm{rank}(S_R)$ or $-t$ (multilayer perceptron)
- Update policy: samples from distribution learned with an autoregressive model
- Many heuristics

Architecture



Main Results

- Improves tensor rank of 4×4 MM over $\mathbb{Z}_{\not\models}$ from 49 to 47
- Rediscovers MM algorithms and DFT
- Improves constant factor for skew-symmetric matrix-vector product
- Finds hardware-specific algorithms

Discussion

- Need discrete F: search for F with ML?
- Border rank, etc.