Course › Week... › 5.5 W... › 5.5.2 ...

5.5.2 Summary

5.5.2 Summary

Discussion

Hide Discussion

Topic: Week 5 / 5.5.2

Add a Post

Show all posts ▼ by recent activity ▼

There are no posts in this topic yet.

✖

Theorem 5.7 *Let $C \in \mathbb{R}^{m \times n}$, $A \in \mathbb{R}^{m \times k}$, and $B \in \mathbb{R}^{k \times n}$. Let*

- $m = m_0 + m_1 + \cdots m_{M-1}$, $m_i \geq 0$ *for* $i = 0, \ldots, M-1$;
- $n = n_0 + n_1 + \cdots n_{N-1}$, $n_j \geq 0$ *for* $j = 0, \ldots, N-1$; *and*
- $k = k_0 + k_1 + \cdots k_{K-1}$, $k_p \geq 0$ *for* $p = 0, \ldots, K-1$.

Partition

$$C = \left(\begin{array}{c|c|c|c} C_{0,0} & C_{0,1} & \cdots & C_{0,N-1} \\ \hline C_{1,0} & C_{1,1} & \cdots & C_{1,N-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline C_{M-1,0} & C_{M-1,1} & \cdots & C_{M-1,N-1} \end{array}\right), A = \left(\begin{array}{c|c|c|c} A_{0,0} & A_{0,1} & \cdots & A_{0,K-1} \\ \hline A_{1,0} & A_{1,1} & \cdots & A_{1,K-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline A_{M-1,0} & A_{M-1,1} & \cdots & A_{M-1,K-1} \end{array}\right),$$

$$\text{and } B = \left(\begin{array}{c|c|c|c} B_{0,0} & B_{0,1} & \cdots & B_{0,N-1} \\ \hline B_{1,0} & B_{1,1} & \cdots & B_{1,N-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline B_{K-1,0} & B_{K-1,1} & \cdots & B_{K-1,N-1} \end{array}\right),$$

with $C_{i,j} \in \mathbb{R}^{m_i \times n_j}$, $A_{i,p} \in \mathbb{R}^{m_i \times k_p}$, *and* $B_{p,j} \in \mathbb{R}^{k_p \times n_j}$. *Then* $C_{i,j} = \sum_{p=0}^{K-1} A_{i,p} B_{p,j}$.

If one partitions matrices C, A, and B into blocks, **and** one makes sure the dimensions match up, **then** blocked matrix-matrix multiplication proceeds exactly as does a regular matrix-matrix multiplication **except** that individual multiplications of scalars commute while (in general) individual multiplications with matrix blocks (submatrices) do not.

Properties of matrix-matrix multiplication

- Matrix-matrix multiplication is *not* commutative: In general, $AB \neq BA$.
- Matrix-matrix multiplication is associative: $(AB)C = A(BC)$. Hence, we can just write ABC.
- Special case: $e_i^T(Ae_j) = (e_i^T A)e_j = e_i^T A e_j = \alpha_{i,j}$ (the i, j element of A).
- Matrix-matrix multiplication is distributative: $A(B+C) = AB + AC$ and $(A+B)C = AC + BC$.

Transposing the product of two matrices

$$(AB)^T = B^T A^T$$

Product with identity matrix

In the following, assume the matrices are "of appropriate size."

$$IA = AI = A$$

Product with a diagonal matrix

$$\left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array}\right) \begin{pmatrix} \delta_0 & 0 & \cdots & 0 \\ 0 & \delta_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_{n-1} \end{pmatrix} = \left(\begin{array}{c|c|c|c} \delta_0 a_0 & \delta_1 a_1 & \cdots & \delta_1 a_{n-1} \end{array}\right)$$

$$\begin{pmatrix} 0 & 0 & \cdots & \delta_{n-1} \end{pmatrix}$$

$$\begin{pmatrix} \delta_0 & 0 & \cdots & 0 \\ 0 & \delta_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_{m-1} \end{pmatrix} \begin{pmatrix} \widetilde{a}_0^T \\ \hline \widetilde{a}_1^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{pmatrix} = \begin{pmatrix} \delta_0 \widetilde{a}_0^T \\ \hline \delta_1 \widetilde{a}_1^T \\ \hline \vdots \\ \hline \delta_{m-1} \widetilde{a}_{m-1}^T \end{pmatrix}$$

Product of triangular matrices

In the following, assume the matrices are "of appropriate size."

- The product of two upper triangular matrices is upper triangular.
- The product of two upper triangular matrices is upper triangular.

Matrix-matrix multiplication involving symmetric matrices

In the following, assume the matrices are "of appropriate size."

- $A^T A$ is symmetric.
- AA^T is symmetric.
- If A is symmetric then $A + \beta x x^T$ is symmetric.

Loops for computing $C := AB$

$$\begin{aligned} C &= \left(\begin{array}{c|c|c|c} \gamma_{0,0} & \gamma_{0,1} & \cdots & \gamma_{0,n-1} \\ \hline \gamma_{1,0} & \gamma_{1,1} & \cdots & \gamma_{1,n-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \gamma_{m-1,0} & \gamma_{m-1,1} & \cdots & \gamma_{m-1,n-1} \end{array}\right) = \begin{pmatrix} \tilde{a}_0^T \\ \hline \tilde{a}_1^T \\ \hline \vdots \\ \hline \tilde{a}_{m-1}^T \end{pmatrix} \left(\begin{array}{c|c|c|c} b_0 & b_1 & \cdots & b_{n-1} \end{array}\right) \\ &= \left(\begin{array}{c|c|c|c} \tilde{a}_0^T b_0 & \tilde{a}_0^T b_1 & \cdots & \tilde{a}_0^T b_{n-1} \\ \hline \tilde{a}_1^T b_0 & \tilde{a}_1^T b_1 & \cdots & \tilde{a}_1^T b_{n-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \tilde{a}_{m-1}^T b_0 & \tilde{a}_{m-1}^T b_1 & \cdots & \tilde{a}_{m-1}^T b_{n-1} \end{array}\right). \end{aligned}$$

Algorithms for computing $C := AB + C$ via dot products.

$$\begin{array}{l} \textbf{for } j = 0, \ldots, n-1 \\ \quad \textbf{for } i = 0, \ldots, m-1 \\ \quad\quad \left.\begin{array}{l} \textbf{for } p = 0, \ldots, k-1 \\ \quad \gamma_{i,j} := \alpha_{i,p}\beta_{p,j} + \gamma_{i,j} \\ \textbf{endfor} \end{array}\right\} \gamma_{i,j} := \tilde{a}_i^T b_j + \gamma_{i,j} \\ \quad \textbf{endfor} \\ \textbf{endfor} \end{array} \quad \text{or} \quad \begin{array}{l} \textbf{for } i = 0, \ldots, m-1 \\ \quad \textbf{for } j = 0, \ldots, n-1 \\ \quad\quad \left.\begin{array}{l} \textbf{for } p = 0, \ldots, k-1 \\ \quad \gamma_{i,j} := \alpha_{i,p}\beta_{p,j} + \gamma_{i,j} \\ \textbf{endfor} \end{array}\right\} \gamma_{i,j} := \tilde{a}_i^T b_j + \gamma_{i,j} \\ \quad \textbf{endfor} \\ \textbf{endfor} \end{array}$$

Computing $C := AB$ by columns

$$\left(c_0 \,\middle|\, c_1 \,\middle|\, \cdots \,\middle|\, c_{n-1} \right) = C = AB = A \left(b_0 \,\middle|\, b_1 \,\middle|\, \cdots \,\middle|\, b_{n-1} \right) = \left(Ab_0 \,\middle|\, Ab_1 \,\middle|\, \cdots \,\middle|\, Ab_{n-1} \right).$$

Algorithms for computing $C := AB + C$:

$$\left.\begin{array}{l} \textbf{for } j = 0, \ldots, n-1 \\ \quad \textbf{for } i = 0, \ldots, m-1 \\ \quad\quad \textbf{for } p = 0, \ldots, k-1 \\ \quad\quad\quad \gamma_{i,j} := \alpha_{i,p}\beta_{p,j} + \gamma_{i,j} \\ \quad\quad \textbf{endfor} \\ \quad \textbf{endfor} \end{array}\right\} c_j := Ab_j + c_j \quad \text{or} \quad \left.\begin{array}{l} \textbf{for } j = 0, \ldots, n-1 \\ \quad \textbf{for } p = 0, \ldots, k-1 \\ \quad\quad \textbf{for } i = 0, \ldots, m-1 \\ \quad\quad\quad \gamma_{i,j} := \alpha_{i,p}\beta_{p,j} + \gamma_{i,j} \\ \quad\quad \textbf{endfor} \\ \quad \textbf{endfor} \end{array}\right\} c_j := Ab_j + c_j$$

$$\textbf{endfor} \qquad\qquad\qquad\qquad \textbf{endfor}$$

Algorithm: $C := \text{GEMM\_UNB\_VAR1}(A, B, C)$

Partition $B \rightarrow \left(B_L \,\middle|\, B_R \right)$, $C \rightarrow \left(C_L \,\middle|\, C_R \right)$
  where B_L has 0 columns, C_L has 0 columns

while $n(B_L) < n(B)$ **do**

  Repartition

$$\left(B_L \,\middle|\, B_R \right) \rightarrow \left(B_0 \,\middle|\, b_1 \,\middle|\, B_2 \right), \left(C_L \,\middle|\, C_R \right) \rightarrow \left(C_0 \,\middle|\, c_1 \,\middle|\, C_2 \right)$$

  where b_1 has 1 column, c_1 has 1 column

$c_1 := Ab_1 + c_1$

  Continue with

$$\left(B_L \,\middle|\, B_R \right) \leftarrow \left(B_0 \,\middle|\, b_1 \,\middle|\, B_2 \right), \left(C_L \,\middle|\, C_R \right) \leftarrow \left(C_0 \,\middle|\, c_1 \,\middle|\, C_2 \right)$$

endwhile

Computing $C := AB$ by rows

$$\left(\begin{array}{c} \tilde{c}_0^T \\ \hline \tilde{c}_1^T \\ \hline \vdots \\ \hline \tilde{c}_{m-1}^T \end{array}\right) = C = AB = \left(\begin{array}{c} \tilde{a}_0^T \\ \hline \tilde{a}_1^T \\ \hline \vdots \\ \hline \tilde{a}_{m-1}^T \end{array}\right) B = \left(\begin{array}{c} \tilde{a}_0^T B \\ \hline \tilde{a}_1^T B \\ \hline \vdots \\ \hline \tilde{a}_{m-1}^T B \end{array}\right).$$

Algorithms for computing $C := AB + C$ by rows:

$$\begin{array}{l} \textbf{for } i = 0, \ldots, m-1 \\ \quad \left.\begin{array}{l} \textbf{for } j = 0, \ldots, n-1 \\ \quad \textbf{for } p = 0, \ldots, k-1 \\ \quad\quad \gamma_{i,j} := \alpha_{i,p}\beta_{p,j} + \gamma_{i,j} \\ \quad \textbf{endfor} \\ \textbf{endfor} \end{array}\right\} \tilde{c}_i^T := \tilde{a}_i^T B + \tilde{c}_i^T \\ \textbf{endfor} \end{array} \quad \text{or} \quad \begin{array}{l} \textbf{for } i = 0, \ldots, m-1 \\ \quad \left.\begin{array}{l} \textbf{for } p = 0, \ldots, k-1 \\ \quad \textbf{for } j = 0, \ldots, n-1 \\ \text{p} \quad\quad \gamma_{i,j} := \alpha_{i,p}\beta_{p,j} + \gamma_{i,j} \\ \quad \textbf{endfor} \\ \textbf{endfor} \end{array}\right\} \tilde{c}_i^T := \tilde{a}_i^T B + \tilde{c}_i^T \\ \textbf{endfor} \end{array}$$

Algorithm: $C := \text{GEMM\_UNB\_VAR2}(A, B, C)$

Partition $A \rightarrow \left(\begin{array}{c} A_T \\ \hline A_B \end{array}\right)$, $C \rightarrow \left(\begin{array}{c} C_T \\ \hline C_B \end{array}\right)$

where A_T has 0 rows, C_T has 0 rows

while $m(A_T) < m(A)$ **do**

Repartition

$$\left(\begin{array}{c} A_T \\ \hline A_B \end{array}\right) \rightarrow \left(\begin{array}{c} A_0 \\ \hline a_1^T \\ \hline A_2 \end{array}\right), \left(\begin{array}{c} C_T \\ \hline C_B \end{array}\right) \rightarrow \left(\begin{array}{c} C_0 \\ \hline c_1^T \\ \hline C_2 \end{array}\right)$$

where a_1 has 1 row, c_1 has 1 row

$$c_1^T := a_1^T B + c_1^T$$

Continue with

$$\left(\begin{array}{c} A_T \\ \hline A_B \end{array}\right) \leftarrow \left(\begin{array}{c} A_0 \\ a_1^T \\ \hline A_2 \end{array}\right), \left(\begin{array}{c} C_T \\ \hline C_B \end{array}\right) \leftarrow \left(\begin{array}{c} C_0 \\ c_1^T \\ \hline C_2 \end{array}\right)$$

endwhile

Computing $C := AB$ via rank-1 updates

$$C = AB = \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{k-1} \end{array} \right) \left(\begin{array}{c} \tilde{b}_0^T \\ \hline \tilde{b}_1^T \\ \hline \vdots \\ \hline \tilde{b}_{k-1}^T \end{array}\right) = a_0 \tilde{b}_0^T + a_1 \tilde{b}_1^T + \cdots + a_{k-1} \tilde{b}_{k-1}^T.$$

Algorithm for computing $C := AB + C$ via rank-1 updates:

$$\begin{array}{l} \textbf{for } p = 0, \ldots, k-1 \\ \quad \left.\begin{array}{l} \textbf{for } j = 0, \ldots, n-1 \\ \quad \textbf{for } i = 0, \ldots, m-1 \\ \quad\quad \gamma_{i,j} := \alpha_{i,p}\beta_{p,j} + \gamma_{i,j} \\ \quad \textbf{endfor} \\ \textbf{endfor} \end{array}\right\} C := a_p \tilde{b}_p^T + C \\ \textbf{endfor} \end{array} \quad \text{or} \quad \begin{array}{l} \textbf{for } p = 0, \ldots, k-1 \\ \quad \left.\begin{array}{l} \textbf{for } i = 0, \ldots, m-1 \\ \quad \textbf{for } j = 0, \ldots, n-1 \\ \quad\quad \gamma_{i,j} := \alpha_{i,p}\beta_{p,j} + \gamma_{i,j} \\ \quad \textbf{endfor} \\ \textbf{endfor} \end{array}\right\} C := a_p \tilde{b}_p^T + C \\ \textbf{endfor} \end{array}$$

Algorithm: $C := \text{GEMM\_UNB\_VAR3}(A, B, C)$

Partition $A \rightarrow \left(A_L \,\middle|\, A_R \right)$, $B \rightarrow \left(\begin{array}{c} B_T \\ \hline B_B \end{array} \right)$

where A_L has 0 columns, B_T has 0 rows

while $n(A_L) < n(A)$ **do**

Repartition

$$\left(A_L \,\middle|\, A_R \right) \rightarrow \left(A_0 \,\middle|\, a_1 \,\middle|\, A_2 \right) , \left(\begin{array}{c} B_T \\ \hline B_B \end{array} \right) \rightarrow \left(\begin{array}{c} B_0 \\ \hline b_1^T \\ \hline B_2 \end{array} \right)$$

where a_1 has 1 column, b_1 has 1 row

$C := a_1 b_1^T + C$

Continue with

$$\left(A_L \,\middle|\, A_R \right) \leftarrow \left(A_0 \,\middle|\, a_1 \,\middle|\, A_2 \right) , \left(\begin{array}{c} B_T \\ \hline B_B \end{array} \right) \leftarrow \left(\begin{array}{c} B_0 \\ \hline b_1^T \\ \hline B_2 \end{array} \right)$$

endwhile