CS 598 EVS: Tensor Computations

Bilinear Algorithms

Edgar Solomonik

University of Illinois, Urbana-Champaign

Bilinear Problems

- A number of basic numerical problems can be thought of as bilinear functions associated with particular order 3 tensors

- These problems admit nontrivial fast *bilinear algorithms*, which correspond to low-rank CP decompositions of the tensors

Bilinear Problems

- A bilinear problem for any inputs $\boldsymbol{a} \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^k$ computes $\boldsymbol{c} \in \mathbb{R}^m$ as defined by a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{m \times n \times k}$
- Variants of discrete convolutions (linear convolution, correlation, cyclic convolution) provide simple examples of $\boldsymbol{\mathcal{T}}$

Bilinear Algorithms

A bilinear algorithm (V. Pan, 1984) $\Lambda = (\boldsymbol{F}^{(A)}, \boldsymbol{F}^{(B)}, \boldsymbol{F}^{(C)})$ computes

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are inputs and $*$ is the Hadamard (pointwise) product.

Bilinear Algorithms as Tensor Factorizations

- A bilinear algorithm corresponds to a CP tensor decomposition

- For multiplication of $n \times n$ matrices, we can define a *matrix multiplication tensor* and consider algorithms with various bilinear rank

Strassen's Algorithm

Strassen's algorithm $\begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{bmatrix}$

$$
\begin{aligned}
\boldsymbol{M}_1 &= (\boldsymbol{A}_{11} + \boldsymbol{A}_{22}) \cdot (\boldsymbol{B}_{11} + \boldsymbol{B}_{22}) \\
\boldsymbol{M}_2 &= (\boldsymbol{A}_{21} + \boldsymbol{A}_{22}) \cdot \boldsymbol{B}_{11} \\
\boldsymbol{M}_3 &= \boldsymbol{A}_{11} \cdot (\boldsymbol{B}_{12} - \boldsymbol{B}_{22}) \\
\boldsymbol{M}_4 &= \boldsymbol{A}_{22} \cdot (\boldsymbol{B}_{21} - \boldsymbol{B}_{11}) \\
\boldsymbol{M}_5 &= (\boldsymbol{A}_{11} + \boldsymbol{A}_{12}) \cdot \boldsymbol{B}_{22} \\
\boldsymbol{M}_6 &= (\boldsymbol{A}_{21} - \boldsymbol{A}_{11}) \cdot (\boldsymbol{B}_{11} + \boldsymbol{B}_{12}) \\
\boldsymbol{M}_7 &= (\boldsymbol{A}_{12} - \boldsymbol{A}_{22}) \cdot (\boldsymbol{B}_{21} + \boldsymbol{B}_{22})
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{C}_{11} &= \boldsymbol{M}_1 + \boldsymbol{M}_4 - \boldsymbol{M}_5 + \boldsymbol{M}_7 \\
\boldsymbol{C}_{21} &= \boldsymbol{M}_2 + \boldsymbol{M}_4 \\
\boldsymbol{C}_{12} &= \boldsymbol{M}_3 + \boldsymbol{M}_5 \\
\boldsymbol{C}_{22} &= \boldsymbol{M}_1 - \boldsymbol{M}_2 + \boldsymbol{M}_3 + \boldsymbol{M}_6
\end{aligned}
$$

By performing the nested calls recursively, Strassen's algorithm achieves cost,

For recent developments in algorithms for fast matrix multiplication, see "Flip Graphs for Matrix Multiplication", Kauers and Moosbauer (2023).

Fast Bilinear Algorithms for Convolution

- Linear convolution corresponds to polynomial multiplication
- The *Toom-Cook* convolution algorithm computes the coefficients of $p \cdot q$ by computing $(p \cdot q)(x_i)$ for $i \in \{1, \dots, n+k-1\}$ and interpolates

Toom-Cook Convolution and the Fourier Transform

- Vandermonde matrices are ill-conditioned with real nodes, but can be perfectly conditioned with complex nodes

- The *fast Fourier transform (FFT)* can be used to perform products with the DFT matrix in $O(n \log n)$ time *Taking $\tilde{\boldsymbol{D}}^{(n)}$ to be the $n_1 \times n_2$ (for $n = n_1 n_2$) leading minor of $\boldsymbol{D}_n$ we can compute $\boldsymbol{y} = \boldsymbol{D}^{(n)}\boldsymbol{x}$ via the split-radix-n_1 FFT,*

$$y_k = \sum_{i=0}^{n-1} x_i \omega_n^{ik} = \sum_{i=0}^{n/2-1} x_{2i}\omega_{n/2}^{ik} + \omega_n^k \sum_{i=0}^{n/2-1} x_{2i+1}\omega_{n/2}^{ik}$$

$$y_{(kn_1+t)} = \sum_{s=0}^{n_1-1} \omega_{n_1}^{st}\left[\omega_n^{sk} \sum_{i=0}^{n_2-1} x_{(in_1+s)}\omega_{n_2}^{ik}\right] \Leftrightarrow \boldsymbol{Y} = ([\tilde{\boldsymbol{D}}^{(n)} \odot (\boldsymbol{D}^{(n_2)}\boldsymbol{A})]\boldsymbol{D}^{(n_1)})^T$$

Cyclic Convolution via DFT

- For linear convolution $\boldsymbol{D}^{(n+k-1)}$ is used, for cyclic convolution $\boldsymbol{D}^{(n)}$ suffices
- The DFT also arises in the eigendecomposition of a circulant matrix

Symmetric Tensor Contractions

- Bilinear algorithms can also be used to accelerate tensor contractions for tensors with symmetry
- Bilinear algorithms for symmetric tensor contractions exist with lower rank than their nonsymmetric counterparts

Symmetric Matrix Vector Product

- Consider computing $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{b}$ with $\boldsymbol{A} = \boldsymbol{A}^T$

Partially-Symmetric Tensor Times Matrix (TTM)

- Can use symmetric mat-vec algorithm to accelerate TTM with partially symmetric tensor from $2n^4$ operations to $(3/2)n^4 + O(n^3)$

Computing Symmetric Matrices

- Output symmetry can also be used to reduced cost, for example when computing a symmetrized outer product $\boldsymbol{C} = \boldsymbol{a}\boldsymbol{b}^T + \boldsymbol{b}\boldsymbol{a}^T$

- To symmetrize product of two symmetric matrices, can compute anticommutator, $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{B}\boldsymbol{A}$

General Symmetric Tensor Contractions

- We can now consider the cost of a symmetrized contraction over v indices of symmetric tensors $\boldsymbol{\mathcal{A}}$ (of order $s+v$) and $\boldsymbol{\mathcal{B}}$ (of order $v+t$)
- Such tensor contractions can be done using $n^{s+t+v}/(s+t+v)! + O(n^{s+t+v-1})$ products

Summary of Bilinear Algorithms

We reviewed bilinear algorithms for 3 problems, which may all be viewed as special cases of tensor contractions

Summary of Nested Bilinear Algorithms

For the tensor $\mathcal{T}^{(n)}$ defining any of the 3 problems for input size n, $\mathcal{T}^{(n)} \otimes \mathcal{T}^{(n)}$ defines a problem for larger inputs