Communication Lower Bounds for Matricized-Tensor Times Khatri-Rao Product

Grey Ballard, Nick Knight, Kathryn Rouse

July 13, 2017

SIAM Annual Meeting
MS76: Communication-Avoiding Algorithms
We apply the communication lower bound approach for general loop nests to a particular tensor computation: matricized-tensor times Khatri-Rao product (MTTKRP)
- establishing lower bounds for sequential and parallel cases

We present optimal algorithms for dense tensors
- separate algorithms for sequential and parallel cases
- they attain the lower bounds to within constant factors

We compare with MTTKRP via matrix multiplication
- new algorithms may perform more computation
- they can perform less communication
An $N^{th}$-order tensor has $N$ modes.

Notation convention: vector $\mathbf{v}$, matrix $\mathbf{M}$, tensor $\mathbf{X}$
A tensor can be decomposed into the fibers of each mode (fibers are vectors – fix all indices but one)
Matricized Tensors

A tensor can be reshaped into a matrix, called a matricized tensor or unfolding, for a given mode, where each column is a fiber.

\[ x = \begin{array}{ccc}
1 & 3 & 7 \\
2 & 4 & 8 \\
5 & 6 & 8
\end{array} \]

\[ x^{(1)} = \begin{bmatrix}
1 & 3 & 5 & 7 \\
2 & 4 & 6 & 8
\end{bmatrix} \]
\[ x^{(2)} = \begin{bmatrix}
1 & 2 & 5 & 6 \\
3 & 4 & 7 & 8
\end{bmatrix} \]
\[ x^{(3)} = \begin{bmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8
\end{bmatrix} \]
CP Decomposition: sum of outer products

Matrix: \[ M \approx \sum_{r=1}^{R} u_r (\sigma_r v_r^T) \]

Tensor: \[ X \approx \sum_{r=1}^{R} u_r \circ v_r \circ w_r \]

This is known as the CANDECOMP or PARAFAC or canonical polyadic or CP decomposition
For fixed rank $R$, we want to solve

$$
\min_{u, v, w} \| \mathbf{X} - \sum_{r=1}^{R} u_r \circ v_r \circ w_r \|
$$

which is a nonlinear, nonconvex optimization problem.

- in the matrix case, the SVD gives us the optimal solution
- in the tensor case, need iterative optimization scheme
Alternating Least Squares (ALS)

Fixing all but one factor matrix, we have a linear LS problem:

$$\min_{\mathbf{V}} \left\| \mathbf{X} - \sum_{r=1}^{R} \mathbf{\hat{u}}_r \circ \mathbf{v}_r \circ \mathbf{\hat{w}}_r \right\|$$

or equivalently

$$\min_{\mathbf{V}} \left\| \mathbf{X}_{(2)} - \mathbf{V}(\mathbf{\hat{W}} \odot \mathbf{\hat{U}})^T \right\|_F$$

\(\odot\) is the Khatri-Rao product, a column-wise Kronecker product.

ALS works by alternating over factor matrices, updating one at a time by solving the corresponding linear LS problem.
Repeat

1. Solve $U(V^T V \ast W^T W) = X_1(W \odot V)$ for $U$

2. Solve $V(U^T U \ast W^T W) = X_2(W \odot U)$ for $V$

3. Solve $W(U^T U \ast V^T V) = X_3(V \odot U)$ for $W$

Linear least squares problems solved via normal equations using identity $(A \odot B)^T(A \odot B) = A^T A \ast B^T B$, where $\ast$ is element-wise product

All optimization schemes that compute the gradient must also compute MTTKRP in all modes
MTTKRP: \[ M = X^{(2)}(W \odot U) \]

Standard approach to MTTKRP for dense tensors
1. “form” matricized tensor (a matrix)
2. compute Khatri-Rao product (a matrix)
3. call matrix-matrix multiplication

We’ll consider alternative approaches that don’t form explicit Khatri-Rao product
MTTKRP for 3-way Tensors

Matrix equation:
\[ M = X_{(2)}(W \odot U) \]

Element equation:
\[ m_{jr} = \sum_{i=1}^{I} \sum_{k=1}^{K} x_{ijk} u_{ir} w_{kr} \]

Example pseudocode:

\[
\begin{array}{c}
\text{for } i = 1 \text{ to } I \text{ do} \\
\quad \text{for } j = 1 \text{ to } J \text{ do} \\
\quad \quad \text{for } k = 1 \text{ to } K \text{ do} \\
\quad \quad \quad \text{for } r = 1 \text{ to } R \text{ do} \\
\quad \quad \quad \quad M(j, r) += X(i, j, k) \cdot U(i, r) \cdot W(k, r)
\end{array}
\]
MTTKRP for N-way Tensors

Matrix equation:

$$M^{(n)} = X^{(n)}(U^{(N)} \odot \ldots \odot U^{(n+1)} \odot U^{(n-1)} \odot \ldots \odot U^{(1)})$$

Element equation:

$$m_{inr}^{(n)} = \sum x_{i_1 \ldots i_N} \prod_{m \neq n} u_{imr}^{(m)}$$

Example pseudocode:

```plaintext
for i_1 = 1 to l_1 do
    . . .
    for i_N = 1 to l_N do
        for r = 1 to R do
            M^{(n)}(i_n, r) += X(i_1, \ldots, i_N) \cdot U^{(1)}(i_1, r) \cdots U^{(N)}(i_N, r)
```
MTTKRP is a set of nested loops that accesses arrays...  
... we just learned how to prove communication lower bounds!

From Nick’s talk...
- tabulate how the arrays are accessed
- use Hölder-Brascamp-Lieb-type inequality in LB proof
- solve linear program to get tightest lower bound
- details in [CDK^+13]
for $i_1 = 1$ to $I_1$ do

\[ \ldots \]

for $i_N = 1$ to $I_N$ do

for $r = 1$ to $R$ do

\[ M^{(n)}(i_n, r) = X(i_1, \ldots, i_N) \cdot U^{(1)}(i_1, r) \cdot \ldots \cdot U^{(N)}(i_N, r) \]

$\Delta =$

\[
\begin{array}{c|cccccc}
& i_1 & \cdots & i_n & \cdots & i_N & r \\
\hline
U^{(1)} & 1 & \cdots & i_n & \cdots & i_N & 1 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
M^{(n)} & & 1 & \cdots & i_N & 1 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
U^{(N)} & & & \ddots & \ddots & \ddots & \ddots \\
X & 1 & \cdots & 1 & \cdots & 1 & 1 \\
\end{array}
\]
Lower bound argument follows [CDK⁺13] almost directly

- solve linear program involving $\Delta$ for tightest bound
Lower bound argument follows [CDK+13] almost directly
- solve linear program involving $\Delta$ for tightest bound

One gotcha: the number of nested loops is not constant
- LB becomes too low by a factor of $O(N)$ (number of loops)
- Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]
Lower bound argument follows [CDK+13] almost directly
- solve linear program involving $\Delta$ for tightest bound

One gotcha: the number of nested loops is not constant
- LB becomes too low by a factor of $O(N)$ (number of loops)
- Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]

Key assumption: algorithm is not allowed to pre-compute and re-use temporary values
- e.g., forming explicit Khatri-Rao product
Lower bound argument follows [CDK+13] almost directly
- solve linear program involving $\Delta$ for tightest bound

One gotcha: the number of nested loops is not constant
- LB becomes too low by a factor of $O(N)$ (number of loops)
- Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]

Key assumption: algorithm is not allowed to pre-compute and re-use temporary values
- e.g., forming explicit Khatri-Rao product

Also used inspiration from memory-independent LBs for matrix multiplication [BDH+12, DEF+13] for parallel case
Theorem

For sufficiently large $I$, any sequential MTTKRP algorithm performs at least

$$\Omega \left( \frac{NIR}{M^{1-1/N}} \right)$$

loads and stores to/from slow memory.

- $N$ is the number of modes
- $l$ is the number of tensor entries
- $R$ is the rank of the CP model
- $M$ is the size of the fast memory
Communication-Optimal Sequential Algorithm (3D)

1. Loop over $b \times \cdots \times b$ blocks of the tensor.
Communication-Optimal Sequential Algorithm (3D)

- Loop over $b \times \cdots \times b$ blocks of the tensor
- With block in memory, loop over subcolumns of input factor matrices, updating corresponding subcolumn of output matrix
- choose $b \approx M^{1/N}$
### Theoretical Comparisons

<table>
<thead>
<tr>
<th>Flops</th>
<th>Lower Bound</th>
<th>New Algorithm</th>
<th>Standard (MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIR</td>
<td>Ω \left( \frac{NIR}{M^{1-1/N}} \right)</td>
<td>O \left( I + \frac{NIR}{M^{1-1/N}} \right)</td>
<td>O \left( I + \frac{IR}{M^{1/2}} \right)</td>
</tr>
<tr>
<td>Words</td>
<td>-</td>
<td>NIR</td>
<td>2IR</td>
</tr>
<tr>
<td>Temp Mem</td>
<td>-</td>
<td>-</td>
<td>\frac{IR}{Tn}</td>
</tr>
</tbody>
</table>
Theoretical Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>New Algorithm</th>
<th>Standard (MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flops</td>
<td>-</td>
<td>$NIR$</td>
<td>$2IR$</td>
</tr>
<tr>
<td>Words</td>
<td>$\Omega \left( \frac{NIR}{M^{1-1/N}} \right)$</td>
<td>$O \left( I + \frac{NIR}{M^{1-1/N}} \right)$</td>
<td>$O \left( I + \frac{IR}{M^{1/2}} \right)$</td>
</tr>
<tr>
<td>Temp Mem</td>
<td>-</td>
<td>-</td>
<td>$\frac{IR}{T_n}$</td>
</tr>
</tbody>
</table>

- New algorithm performs $N/2$ more flops than standard
- For relatively small $R$, $I$ term dominates communication
  - we expect this to be the typical case in practice
- For relatively large $R$, new algorithm communicates less
  - better exponent on $M$
Any parallel MTTKRP algorithm involving a tensor with \( I_k = I^{1/N} \) for all \( k \) and that evenly distributes one copy of the input and output performs at least

\[
\Omega \left( \left( \frac{NIR}{P} \right)^{\frac{N}{2N-1}} + NR \left( \frac{I}{P} \right)^{1/N} \right)
\]

sends and receives. (Either term can dominate.)

- \( N \) is the number of modes
- \( I \) is the number of tensor entries
- \( I_k \) is the dimension of the \( k \)th mode
- \( R \) is the rank of the CP model
- \( P \) is the number of processors
Communication-Optimal Parallel Algorithm (3D)

Each processor
1. Starts with one subtensor and subset of rows of each input factor matrix
Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
Communication-Optimal Parallel Algorithm (3D)

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix

2. All-Gathers all the rows needed from $U^{(1)}$

3. All-Gathers all the rows needed from $U^{(3)}$

4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
5. Reduce-Scatters to compute and distribute $M^{(2)}$ evenly
### Theoretical Comparisons

<table>
<thead>
<tr>
<th>Words (&quot;small&quot; $P$)</th>
<th>Lower Bound</th>
<th>New Algorithm</th>
<th>Standard (MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega \left( NR \left( \frac{I}{P} \right)^{1/N} \right)$</td>
<td>$O \left( NR \left( \frac{I}{P} \right)^{1/N} \right)$</td>
<td>$O \left( I^{1/N} R \right)$</td>
<td></td>
</tr>
</tbody>
</table>

- For relatively small $P$ (or small $R$) and even dimensions, parallel algorithm attains lower bound.
- Comparison with matrix multiplication from [DEF⁺13]
  - ignores parallel cost of forming Khatri-Rao product.
### Theoretical Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>New Algorithm</th>
<th>Standard (MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Words</strong> (&quot;small&quot; $P$)</td>
<td>$\Omega \left( NR \left( \frac{I}{P} \right)^{1/N} \right)$</td>
<td>$O \left( NR \left( \frac{I}{P} \right)^{1/N} \right)$</td>
<td>$O \left( I^{1/N} R \right)$</td>
</tr>
<tr>
<td><strong>Words</strong> (&quot;large&quot; $P$)</td>
<td>$\Omega \left( \left( \frac{NIR}{P} \right)^{\frac{N}{2N-1}} \right)$</td>
<td>$O \left( \left( \frac{NIR}{P} \right)^{\frac{N}{2N-1}} \right)$</td>
<td>$O \left( \left( \frac{IR}{P} \right)^{2/3} \right)$</td>
</tr>
</tbody>
</table>

- For relatively small $P$ (or small $R$) and even dimensions, parallel algorithm attains lower bound.
- Comparison with matrix multiplication from [DEF+13]
  - ignores parallel cost of forming Khatri-Rao product
- For larger $P$ (or $R$), then we need different algorithm
  - also parallelize over columns of output matrix
  - involves communicating the tensor
We apply the communication lower bound approach for general loop nests to a particular tensor computation: matricized-tensor times Khatri-Rao product (MTTKRP)  
  establishing lower bounds for sequential and parallel cases

We present optimal algorithms for dense tensors  
  separate algorithms for sequential and parallel cases  
  they attain the lower bounds to within constant factors

We compare with MTTKRP via matrix multiplication  
  new algorithms may perform more computation  
  they can perform less communication
What about lower bounds for...

- all $N$ MTTKRP\$s?
  - in sequence, as in CP-ALS
  - all at once, as in gradient-based methods
- partial MTTKRP\$s and multi-TTV\$s?
  - MTTKRP methods that violate key assumption
  - [PTC13, KU16, LCP$^+$17]
- other tensor computations like tensor-times-matrix?
  - useful for Tucker decomposition

We believe the same lower bound framework will work, still working out the details
Brief announcement: strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds. 

Communication lower bounds and optimal algorithms for programs that reference arrays - part 1. 

Communication-optimal parallel recursive rectangular matrix multiplication. 

Oguz Kaya and Bora Uçar. 
Parallel CP decomposition of sparse tensors using dimension trees. 

J. Li, J. Choi, I. Perros, J. Sun, and R. Vuduc. 
Model-driven sparse CP decomposition for higher-order tensors. 
In IEEE International Parallel and Distributed Processing Symposium, IPDPS, pages 1048–1057, May 2017.

Anh-Huy Phan, Petr Tichavsky, and Andrzej Cichocki. 
Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations. 

Tyler Michael Smith and Robert A. van de Geijn. 
Pushing the bounds for matrix-matrix multiplication. 