THE STRUCTURE AND RANK OF MxPxQ TENSORS: THE HEURISTIC APPROACH

Sharon J. Laskowski & David P. Dobkin

Research Report #130

April 1978

An earlier version of this paper was presented at the Conference on Information Science and Systems, March 1978. This research was supported in part by NSF grants MCS 76-11460, MCS 74-24193, and DCR 74-12870.

Abstract

A new approach to tensor ranking, that of implementing a set of heuristics to automatically construct rankings is presented. The exact rank of any 2xpxq tensor over Z_2 and near optimal ranks of mxpxq tensors can be calculated via this method. Furthermore, the success of these automated heuristics points out the value of such an experimental tool in offering the researcher new insights into a difficult mathematical problem.

1. Introduction

Tensors (or 3-way arrays) and the tensor ranking problem occur in such contexts as arithmetic complexity [4], the statistical methods of individual difference scaling [9], and in problems of data compression [11]. For the purposes of this paper, a tensor can be thought as a three dimensional array or, more specifically, a set of m p by q matrices over a field K. In this paper K will be Z_2, the field of integers mod 2. Tensor rank is a generalized version of the rank of a matrix but with a decomposition into a minimum linear combination of rank 1 tensors instead of the matrix decomposition into a linear combination of rank 1 matrices.

Unlike the two dimensional case, however, there is no efficient, constructive procedure for calculating the rank of an arbitrary tensor. This problem is shown to be mathematically difficult by the recent work of de Groote [6]. Decidability is only known over algebraically closed fields (by the super-exponential Tarski decision procedure [16]) and over finite fields (by the exponential enumeration procedure). In the past, specific tensors have been ranked through careful, yet *ad hoc* examination of their structure. The techniques which have been applied have combined sophisticated ideas from linear algebra as well as trial-and-error methods designed for the tensor being studied. For example, the tensor representing matrix multiplication has been analyzed in detail and the exact complexity of the 2 by 2 matrix case is known, although the upper and lower bounds for the 3 by 3 case remain open with the tensor rank lying somewhere between 18 and 23. These approaches usually do not aid in finding efficient, constructive solutions for large classes of tensors.

A different method for tensor ranking is described here. By observing the structure of "small" tensors and the role they play in the structure of more complicated tensors, a set of heuristics is formulated. When automated, these heuristics can calculate ranks of small tensors exactly and can be applied to larger tensors resulting in ranks that are reasonably close to exact in many instances.

The immediate advantage to this formulation is that a computer implementation allows the heuristics to be applied to problems too large to solve by hand calculation. In addition, the results of the "experimental" applications give a broader sense of the power of these heuristics and feedback as to where they fail and how they can be improved. Apart from the ranking problem this approach has merit in its own right as a new methodology for solving a hard mathematical problem whose solution in the general case has thus far remained intractable. Since these heuristics are designed to determine, in low degree polynomial time, an exact or approximate solution to a problem for which all known solutions require exponential time, they can be viewed as belonging to the class of methods for approximating NP-complete problems [8].

Section 2 places the ranking problem in the proper context and describes the implementation and application of the heuristics for constructing upper bounds. A brief algebraic coding theory background and the lower bound theorem is presented in section 3. Section 4 describes a normal form in which all 2xpxq tensors over Z_2 field can be represented. The heuristics, when applied to these tensors, are shown to generate exact upper bounds. Section 5 discusses the implications of these results and further directions to explore.

2. The Heuristics and the Implementation

2.1 The Model

The tensor model has been studied in detail by arithmetic complexity theorists such as Brockett and Dobkin [2],[4], Fiduccia [5], and Hopcroft and Musinski [7] as a means of classifying bilinear form multiplication problems. The notation here is that of Dobkin [4].

An mxpxq third order tensor is a set of m p by q matrices $\{G_i\}$, $1 \leq i \leq m$. This set can be characterized in two dimensions by a degree one matrix polynomial in m indeterminants called G(s).

Definition $G(s) = \sum_{i=1}^{m} s_i G_i$ where $\{G_i\}$ is a set of matrices and the s_i's are indeterminants functioning as placeholders.

The G_i matrices are called the basis matrices and G(s) the characteristic matrix of the problem. If a, b, and c are vectors whose elements are members of Z_2, the tensor product $a \otimes b \otimes c$ with i,j,kth element $a_i b_j c_k$ is a rank 1 tensor or dyad. The rank of G(s) is the minimum d for which G(s) is expressible as the sum of d dyads. Any decomposition of G(s) into t dyads is called a realization of G(s) of dimension t.

It is important to note that the tensors studied here have been limited to the field Z_2, the integers mod 2. Of course, any lower bound over Z_2 is a lower bound over the integers, and often an upper bound over Z_2 can be converted to one over the integers by inserting the appropriate + and - signs. A simple example of a tensor which represents polynomial multiplication is:

$$G_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad G_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad G_3 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{so } G(s) = \begin{bmatrix} s_1 & s_2 \\ s_2 & s_3 \end{bmatrix}$$

A naive realization of G(s) consists of 4 dyads (one for each position in G(s)), but a realization of 3 dyads is possible and as the lower bound theorem will show later, this bound is exact:

$$\begin{bmatrix} s_1+s_2 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} s_2 & s_2 \\ s_2 & s_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & s_2+s_3 \end{bmatrix}$$

2.2 The Heuristics

The ideas developed for the tensor ranking programs have their foundation in what may be loosely termed an artificial intelligence approach, the simulation of intelligent behavior in solving a problem. However, the tensor ranking system, once able to imitate human behavior, is intended to go beyond human capabilities, ranking tensors that have previously been to large to conceptualize. This approach takes its motivation from heuristic programming as presented in Slagle [15], symbolic and algebraic manipulation [12], [13], including such systems as MACSYMA, and automatic numerical analysis [14], with its software for solving ordinary differential equations and root-finding, for example. Taken in this spirit, a tensor ranking system can be a experimental tool used by the researcher to aid in finding new insights into the ranking problem.

The heuristics have been implemented as a set of programs which construct Z_2 realizations for tensors as close to exact as possible with the knowledge of tensor structure the programs have been given. The System of Programs for Tensor Ranking (SPTR) is based on three simple heuristics:

(1) A large tensor can be divided into subtensors called building blocks. These building blocks are "small" tensors for which a known exact realization is better than the naive realization.

(2) When the tensor is written as a sum of these blocks and if the blocks are chosen and overlapped properly, its rank will be equal to the sum of the ranks of the building blocks minus any savings from overlaps.

(3) For the overlapping to be correct, the building blocks often must be applied in a certain order.

For example if we consider the tensor

$$\begin{bmatrix} s_1 & s_2 \\ s_2 & s_3 \end{bmatrix}$$

to be a building block, it can be applied to the following tensor 3 times to produce an exact ranking of 6 dyads:

$$\begin{bmatrix} s_1 & s_2 & s_3 \\ s_2 & s_3 & s_4 \\ s_3 & s_4 & s_5 \end{bmatrix} = \begin{bmatrix} s_2 & s_2 & 0 \\ s_2 & s_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & s_4 & s_4 \\ 0 & s_4 & s_4 \end{bmatrix} + \begin{bmatrix} s_3 & 0 & s_3 \\ 0 & 0 & 0 \\ s_3 & 0 & s_3 \end{bmatrix}$$

$$+ \begin{bmatrix} s_1+s_2+s_3 & 0 & 0 \\ 0 & s_2+s_3+s_4 & 0 \\ 0 & 0 & s_3+s_4+s_5 \end{bmatrix}$$

SPTR consists of a lower bound program, simplification programs, a library of building blocks and a driver program all running in polynomial time.

The lower bound program generates a bound which gives some measure of how exact the upper bounds are. The simplification programs put the tensor into a less complicated form via several elimination procedures. The driver program calls on the library to apply appropriate building blocks. Surprisingly, the building block library has remained small.

By experimenting with SPTR a good understanding of the structure of tensors has developed. The results for ranking 2xpxq tensors are presented here since they are exact. It is easy to see that these methods can be generalized to arbitrary mxpxq tensors. In fact, SPTR will accept any size tensor.

2.3 The Building Block Library

Only building blocks for the 2xpxq case will be considered here since the proofs of exact bounds in this paper are restricted to this size. There are four building blocks required to rank all 2xpxq tensors. their basic forms, up to permutations of the rows and columns, and decompositions into dyads are:

U1) $$\begin{bmatrix} s_1 & s_2 & 0 \\ 0 & s_1 & s_2 \end{bmatrix} = \begin{bmatrix} s_1 & s_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & s_2 & s_2 \end{bmatrix} + \begin{bmatrix} 0 & s_1+s_2 & 0 \\ 0 & s_1+s_2 & 0 \end{bmatrix}$$

U2) $$\begin{bmatrix} s_1 & s_2 \\ 0 & s_1+s_2 \end{bmatrix} = \begin{bmatrix} s_1 & s_1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & s_1+s_2 \\ 0 & s_1+s_2 \end{bmatrix}$$

U3) $$\begin{bmatrix} s_1 & s_2 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_1+s_2 & s_2 \\ 0 & 0 & 0 & s_1+s_2 \end{bmatrix} = \begin{bmatrix} 0 & s_2 & 0 & s_2 \\ 0 & 0 & 0 & 0 \\ 0 & s_2 & 0 & s_2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} s_1 & 0 & 0 & s_1 \\ 0 & s_1 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & s_1+s_2 \\ 0 & 0 & 0 & 0 \\ 0 & s_1+s_2 & s_1+s_2 & 0 \\ 0 & 0 & 0 & s_1+s_2 \end{bmatrix}$$

U4) $\begin{bmatrix} s_2 & \ldots\ldots & s_2 \end{bmatrix}$ = 1 dyad

3. The Lower Bound Theorem

The lower bounds originate from results in algebraic coding theory on the structure of 0-1 matrices. In [3] and [4] these ideas have been exploited to yield some crude general lower bounds and slightly better bounds for some specific tensors. A finer tuning for the lower bounds is presented here and it plays an important role in showing the upper bounds in section 4 to be exact.

A few definitions are needed at this point. For any realization of a tensor, a "code" of {0,1} vectors can be constructed. Let v_i, $1 \leq i \leq m$, be {0,1} vectors. $v_i \oplus v_j$ is a vector whose kth entry is the exclusive OR of the kth entries in v_i and v_j. $|v_i|$ is the Hamming norm, (number of 1's in v_i). A <u>code</u> consists of m v_i's each having the length l. Given a realization of l dyads a corresponding code can be constructed. The kth entry of v_i is 1 if and only if the kth dyad includes an s_i and is 0 otherwise. If G_i has matrix rank k (recalling that G_i has a 1 wherever an s_i occurs in the tensor), then $|v_i|$ must be $\geq$ k. Similarly, if $G_i \oplus G_j$ has rank k, then $|v_i \oplus v_j| \geq k$, and so on for all possible sums of G_i's. All possible exclusive OR sums over the v_i's are called the <u>constraints</u> of G(s).

The code for the previously mentioned building block is:

v_1 =	1 0 0	s_1
v_2 =	1 1 1	s_2
v_3 =	0 0 1	s_3

Exploiting some ideas from algebraic coding theory leads to a lower bound which is easily calculable. As was mentioned previously, any realization can be represented by a code and hence a lower bound on the rank consists of calculating the constraints on the v_i's and constructing the minimum length code. Any realization for a tensor G(s) has to have at least as many dyads as that length.

<u>Theorem</u> Let G(s) be an mxpxq tensor over Z_2.

$$\text{The rank of } G(s) \geq \Big(\sum_{I \subseteq \{1,\ldots,m\}} \Big| \bigoplus_{i \text{ in } I} v_i \Big| \Big) / 2^{m-1}$$

Proof: Let w be a vector with jth entry the ith entry of v_j. Then,

$$\sum_{I \subseteq \{1,\ldots,m\}} \Big| \bigoplus_{i \text{ in } I} w_i \Big|$$

is 2^{m-k} times the number of subsets of {1,...,k} with an odd number of elements where $k=|w|$. This quantity is $2^{m-k}\,2^{k-1}=2^{m-1}$. If $v_1, \ldots, v_m$ is a minimum length code for G(s), its length multiplied by 2^{m-1} must be equal to the sum of all the constraints since the sum over one entry of each of the v_i's is 2^{m-1}. The theorem follows immediately.

This theorem provides a concise way of calculating lower bounds. As will be shown in section 4, the lower bound is exact in the 2xpxq case except for instances of the following subtensors where the bound is one dyad less than the rank: (Note that substituting s_1+s_2 for s_1 in a pattern is considered to be that same pattern.)

L1) $\begin{bmatrix} s_1 & s_2 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_1 & s_2 \\ 0 & 0 & 0 & s_1 \end{bmatrix}$ L2) $\begin{bmatrix} s_1(+s_2) & s_2 & 0 & 0 \\ 0 & 0 & s_1(+s_2) & s_2 \end{bmatrix}$

4. <u>The Upper Bound Theorem</u>

Throughout this section it is assumed that the matrix rank of some combination of the G_i's is n for a 2xnxn tensor. Without loss of generality, the rank of G_1 will be equal to n n since the indetermi-nants can always be renamed.

In order to rank any tensor with the building blocks U1-U4 the tensor must have a particular structure.

<u>Definition</u> A 2xnxn tensor is in <u>normal form</u> if it has the following structure:

1. G_1 is the identity matrix.

2. $(G_2)_{ij}$ is 0 unless

   i. $i=j$

   ii. $i+1=j$

   iii. $i>j$ and $(G_2)_{ik}$ $(k\neq j,i) = 0$ and $(G_2)_{kj}$ $(k\neq j,i) = 0$

   or iv. $i>j$ and $(G_2)_{i,i+1}=1$, $(G_2)_{j,j+1}=1$

3. No kxk subtensor has the form $G_1=I$ and $(G_2)_{ij}=(1$ if $i+1=j(\text{mod}k)$ and 0 otherwise + possibly I).

A typical example of a tensor in normal form is:

$$\begin{bmatrix} s_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & s_1+s_2 & s_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_1+s_2 & s_2 & 0 & 0 \\ 0 & s_2 & 0 & 0 & s_1+s_2 & s_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & s_1 & 0 \\ 0 & 0 & 0 & s_2 & 0 & 0 & s_1 \end{bmatrix}$$

<u>Lemma 1</u> Any 2xnxn tensor can be put into normal form with no change in rank.

The proof is sketched here. Further details will appear in [10].

Proof: Let G(s) be a 2xnxn tensor. It is easy to see that if a row (column) is replaced by the sum of that row (column) and another, the rank of G(s) will not change. The following procedure puts G(s) into normal form.

1. Use Gaussian elimination to put only s_1 $(+s_2)$'s on the diagonal. (structures 1. and 2i.)

2. Eliminate the extra s_2's above the diagonal one at a time by adding rows down and columns across to remove any extra s_1's which were introduced and vice versa. By rearranging rows the s_2's can be

placed at positions $(G_2)_{i,i+1}$. (2ii.)

3. Remove the s_2's below or across from an s_2 at location i,i+1. (2iii. and 2iv.)

4. If there is more than 1 s_2 in a row or column below the diagonal, all but one to a row or column can be removed except for cases handled by 6. (2iii.)

5. The kxk exception requires a procedure of adding rows 1 through k-1 to k and column k to columns 1 through k-1, and repeating 2, 3. and 4. (3.)

6. Certain cases with an extra s_2 in column(row) must be treated specially. Procedures 1-5 will not work for the following tensor structure:

$$\begin{bmatrix} s_1(+s_2) & s_2 & 0 & 0 \\ 0 & s_1(+s_2) & 0 & 0 \\ s_2 & 0 & s_1+(s_2) & s_2 \\ s_2 & 0 & 0 & s_1(+s_2) \end{bmatrix}$$

In general this is simplified by adding the first two rows down, the appropriate columns across, and then applying 3-4 if necessary. (2.iii)

<u>Lemma 2</u> Any 2xnxn tensor G(s) can be ranked exactly with the four building blocks U1 through U4.

proof: Assume G(s) is in the normal form.

The upper bound is calculated by first applying U4 across the rows of G(s). Then building blocks U1, U2, and U3 are located in G(s) followed by L1 and L2. Suppose there are X U4's occurring as:

$$\begin{bmatrix} \dots & s_1 & s_2 & 0 \\ s_2 & \dots & s_1 & s_2 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \dots & s_1+s_2 & s_2 & 0 \\ s_2 & \dots & s_1+s_2 & s_2 \end{bmatrix}$$

and there are U5 s_1's and U6 s_1+s_2's not contained by any of the other patterns, and R remaining s_2's, then an upper bound on the rank of G(s) is:

$$3U1 + 2U2 + 5U3 + U4 - X + U5 + U6 + 6L1 + 4L2 + R$$

Note that the names of the patterns are representing the number of them located for these calculations.

To calculate the lower bound the information the building blocks supply is substituted for the matrix ranks in the lower bound formula:

lower bound of G(s) =

$(|v_1|+|v_2|+|v_1 \oplus v_2|)/2$ + patterns not matching lower bound formula =

$$(6U1+4U2+10U3+2U4+2U5+2U6+10L1+6L2+2R-2X)/2 + L1 + L2$$

$$= 3U1 + 2U2 + 5U3 + U4 + U5 + U6 + 6L1 + 4L2 + R - X$$

and the upper bound is exact.

<u>Theorem</u> Any 2xpxq tensor G(s) can be ranked exactly with the four building blocks U1 through U4.

Proof: Suppose $|v_1|=n\leq p\leq q$. G(s) can be split into an n by n tensor in normal form bordered on the right and bottom by q-n columns and p-n rows of s_2's which are linearly independent. By first applying U4 across and down and proceeding as in Lemma 2, the tensor will be ranked exactly.

5. Discussion

Further research [10] is being undertaken to expand the building block library so that SPTR can be applied to more complex structures normally impossible to study. Thus far, the class of 2xpxq tensors has been analyzed exactly with the help of SPTR, but this is not to neglect other tensors. Experimental results imply that SPTR has the potential for generating new information about larger classes of tensors as well as specific tensors. SPTR at this point performs just as well if not better than known methods for m slightly larger than 2 and, for example, with the addition of one building block SPTR was able to find the algorithm for multiplying 3 by 3 matrices in 23 multiplications.

SPTR's building block library and simplification programs are being expanded in order to further examine three aspects of such an approach to the tensor ranking problem. First, there are certain tensors for which SPTR can generate exact bounds and it is interesting to see what characteristics makes a tensor "easy" to rank. Second, the system should always runs efficiently, when producing bounds, whether exact or approximate, in fact, the procedures described here run on the order of n^3 time. This characteristic suggests the third aspect. The researcher is aided in the search for insights into the structure of large classes of tensors by experimental results calculated by SPTR. This interaction speeds up the testing of new hypotheses and often suggests new conjectures which would not ordinarily be discovered. In a more global sense, these results illustrate the validity of approaching a difficult mathematical problem from the perspective of automating and formalizing heuristics for searching out and verifying solutions to instances of the problem.

References

[1] Elwyn R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.

[2] Roger W. Brockett, & David P. Dobkin, "On the Optimal Evaluation of a Set of Bilinear Forms", J. of Linear Algebra, to appear.

[3] Mark R. Brown & David P. Dobkin, "An Improved Lower Bound on Polynomial Multiplication", unpublished manuscript, 1977.

[4] David P. Dobkin, "On the Arithmetic Complexity of a Class of Arithmetic Computations", Ph.D. thesis, Harvard University, Sept. 1973. (Also, Yale University Computer Science Technical Report 23, Oct. 1973.)

[5] Charles M. Fiduccia, "On Obtaining Upper Bounds on the Complexity of Matrix Multiplication", in Complexity of Computer Computations, (Miller & Thatcher, Editors), Plenum Press, 1972.

[6] Hans F. de Groote, "On Varieties of Optimal Algorithms for the Computation of Bilinear Mappings, Mathematisches Institut der Universität Tübingen technical report, 1978.

[7] John Hopcroft & Jean Musinski, "Duality Applied to the Complexity of Matrix Multiplication and Other Bilinear Forms", SIAM J. Com-

puting, Vol. 2, Sept. 1973.

[8] David S. Johnson, "Approximation Algorithms for Combinatorial Problems", J. of Computer and System Sciences, 9, 1974.

[9] Joseph B. Kruskal, "Trilinear Decomposition of Three-way Arrays: Rank and Uniqueness in Arithmetic Complexity and in Statistical Models", J. of Linear Algebra, to appear.

[10] Sharon J. Laskowski, "On Heuristics for Tensor Ranking", Ph.D. thesis, Yale University, to appear.

[11] Richard J. Lipton & Robert Tuttle, private communication.

[12] Joel Moses, "Algebraic Simplification: A Guide for the Perplexed", CACM, vol., 14, Aug., 1971.

[13] Joel Moses, "Symbolic Intergration, the Stormy Decade", CACM, vol. 14, Aug., 1971.

[14] John R. Rice, editor, Mathematical Software, Academic Press, 1971.

[15] James R. Slagle, Artificial Intelligence: the Heuristic Programming Approach, McGraw Hill Book Company, 1971.

[16] A. Tarski, A Decision Method for Elementary Algebra and Geometry, 2nd edition, revised, Berkeley and Los Angeles, 1951.