Tensor Rank is Hard to Approximate

Joseph Swernofsky[1]
Kungliga Tekniska Högskolan, Lindstedtsvägen 3, Stockholm SE-100 44, Sweden
josephsw@kth.se

Abstract

We prove that approximating the rank of a 3-tensor to within a factor of $1 + 1/1852 - \delta$, for any $\delta > 0$, is NP-hard over any field. We do this via reduction from bounded occurrence 2-SAT.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases tensor rank, high rank tensor, slice elimination, approximation algorithm, hardness of approximation

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2018.26

Acknowledgements I thank Johan Håstad for guidance during research and for thoroughly reviewing and giving advice on the writing of this paper. I thank Per Austrin for discussions and for improving the inapproximability constant.

1 Introduction

The rank of a matrix is well understood and easy to compute. The student first introduced to the notion of rank often learns that they can perform Gaussian elimination on a matrix and the number of nonzero rows they obtain is its rank. They may even learn that this is equivalent to the number of linearly independent rows of the matrix. A rank 1 matrix can be written as an outer product of two vectors u and v, meaning its ij^{th} entry is $u_i \cdot v_j$. The rank of M is equivalent to the minimum number of outer products (rank 1 matrices) that must be added to form M.

This notion of rank has been generalized to describe tensors. A 3-tensor T is a grid of numbers with 3 indices. A tensor is rank 1 if it is the outer product of 3 vectors, and the rank of T is the minimum number of rank 1 tensors that must be added to form it.

Rank of 3-tensors is, apart from its inherent mathematical appeal, a natural tool for reasoning about bilinear circuit complexity. The inputs and outputs of a bilinear circuit correspond to indices in the 3 coordinates of a 3-tensor, and the rank is equal to the number of multiplication nodes needed in the circuit [8]. Strassen's [17] famous algorithm for 2×2 matrix multiplication can be viewed in this light. The multiplication is a circuit computing the entries of the output matrix from the entries of the input matrices. Writing this as a $2^2 \times 2^2 \times 2^2$ tensor, his basic construction corresponds to a rank 7 decomposition. He then applied this recursively to create the first algorithm for $n \times n$ matrix multiplication that was faster than the naive algorithm.

Unfortunately, while matrix rank is easy to compute, Håstad [6] showed that tensor rank is NP-hard to calculate. More recently, Shitov [14] and Schaefer and Štefankovič [13] showed that rank over a field $\mathbb{F}$ is complete for the existential theory of $\mathbb{F}$ and is also uncomputable over $\mathbb{Z}$. This presents a roadblock for researchers hoping to analyze the rank of tensors for

[1] Supported by the Knut and Alice Wallenberg Foundation.

specific applications. Much work has been done since Strassen's initial paper to analyze larger tensors modeling matrix multiplication and many improvements have been obtained. However, even very simple tensors stubbornly resist analysis, such as the $3^2 \times 3^2 \times 3^2$ tensor for 3×3 matrix multiplication. Its rank is only known to be between 19 [3] and 23 [9].

Another application for tensor rank is demonstrated by Raz [12], who shows how strong circuit lower bounds for more general circuits may be possible with a better understanding of rank. Tensor rank is also a practical tool in a variety of fields, such as signal processing and computer vision [7]. Our knowledge is very limited though, because despite knowing that there are 3-tensors with rank at least $n^2/3$ (which can be seen by dimension counting), the highest rank known for an explicit family of 3-tensors is $O(n)$ [15]. Also, while we know rank is hard to compute exactly, we do not know if it can be approximated. Alexeev et al. [1] and Bläser [4] both mention this open question. Recently Song et al. [16] proved that, assuming the Exponential Time Hypothesis (ETH), there is some constant c_0 so that tensor rank cannot be approximated within a factor of c_0 in polynomial time. We strengthen this to prove an NP-hardness result and in particular the following theorem.

▶ **Theorem 1.** *It is NP-hard to approximate 3-tensor rank over any field $\mathbb{F}$ within a factor of $1 + 1/1852 - \delta$, for any $\delta > 0$.*

Our construction is a re-analysis and simplification of the reduction by Håstad [6]. We show that if bounded occurrence SAT is used as the starting point for the reduction then significant extra rank can be guaranteed from unsatisfied clauses over the rank in the satisfiable case. Independently, Bläser et al. [5] proved the same result, but without an explicit constant, and with a slightly more involved argument.

A natural follow-up question to ask is whether tensor rank is hard to approximate within any constant, or within a specific unbounded function. It would also be interesting to have any nontrivial approximation algorithm. We discuss these questions briefly in Section 4.

2 Preliminaries

We work over an arbitrary field $\mathbb{F}$ throughout this paper.

A ***d*-tensor** is a function from d-tuples of natural numbers to $\mathbb{F}$. The entries in the tuple represent coordinates in a d-dimensional space. A given d-tensor has size n_i in coordinate i and is defined at exactly those s where $s_i \in [n_i]$ for all i. It is helpful to think of this as a grid with numbers written in the cells. We typically write T_s to denote $T(s)$, or even $T_{s_1 s_2 \ldots s_d}$. For $v_i \in \mathbb{F}^{n_i}$, an outer product $\otimes_{i \in [d]} v_i$ of d vectors is a rank 1 tensor. The **rank, $\mathbf{rk}(\boldsymbol{T})$,** is the minimum r so that $T = \sum_{j \in [r]} \otimes_{i \in [d]} v_i^j$.

A minimization problem P consists of a set of valid strings L together with a score function $s : L \to \mathbb{R}^+$. An **approximation algorithm** for P is an algorithm A that takes strings in L and outputs numbers in $\mathbb{R}^+$ with $\forall x \in L. A(x) \geq s(x)$. Let L_n be the strings in L of length n. For a function $c : \mathbb{N} \to \mathbb{R}^+$, A is a c-approximation algorithm if $\max_{x \in L_n} \frac{A(x)}{s(x)} \leq c(n)$.

A **MAX-E2-SAT** instance consists of a set of variables $\mathbf{x} = \{x_k\}_{k \in [n]}$ and a set of disjunctive clauses $\{l_{i1} \vee l_{i2}\}_{i \in [m]}$ of size exactly 2 over those variables. Here $n, m \in \mathbb{N}$ and l_{ij} is a literal of a variable in $\mathbf{x}$ for each $i \in [m], j \in [2]$. The problem is to compute the maximum number of clauses that can be simultaneously satisfied by an assignment to $\mathbf{x}$. **E3-OCC-MAX-2SAT** is MAX-E2-SAT where every variable occurs in exactly 3 clauses. Note that in E3-OCC-MAX-2SAT we have $m = \frac{3n}{2}$.

2.1 Slices

A **slice** of a tensor is the grid obtained from fixing one of the coordinates. The slice $T_{i:x}$ of a d-tensor T is a $(d-1)$-tensor with $T_{i:x}(s) := T(s_1, s_2, \dots s_{i-1}, x, s_i, \dots s_{d-1})$. That is, when indexing into $T_{i:x}$ one is indexing into T but omitting the coordinate i. For example, if $d = 3$ then $T_{3:k}$ is the matrix M where $M_{ij} = T_{ijk}$. Even more concretely, if we draw the first coordinate as the row and the second coordinate as the column:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, A_{2:1} = \begin{bmatrix} a \\ c \end{bmatrix}$$

We want to recover some of our intuition about matrix rank and use row reduction when computing tensor rank.

▶ **Lemma 2.** *If a slice is scaled by a nonzero constant or added to another slice in the same coordinate, the rank does not change.*

Proof. Suppose we have a rank r decomposition $T = \sum_{h\in[r]} \otimes_{j\in[d]} v_j^h$. If slice $T_{i:x}$ is scaled by λ, simply replace $v_i^h(x)$ with $\lambda v_i^h(x)$ for every h. This shows that the rank does not increase. Given a rank r' decomposition of the new tensor, scale by $1/\lambda$ to see that $r' \geq r$.

To add $T_{i:x}$ to $T_{i:y}$, simply replace $v_i^h(y)$ with $v_i^h(y) + v_i^h(x)$. Again, this operation can be inverted by replacing $v_i^h(y)$ with $v_i^h(y) - v_i^h(x)$, so the ranks before and after adding $T_{i:x}$ to $T_{i:y}$ are the same. ◀

▶ **Corollary 3.** *If a slice is a linear combination of other slices in the same coordinate, setting every entry to 0 or removing it does not change the rank.*

We call the process of iteratively removing dependent slices until all remaining slices are independent **slice elimination**. When performing row reduction on a matrix, one can add any row u to any other row v without changing the rank. Gaussian elimination turns this into a simple strategy for calculating rank where a row u is added, scaled appropriately, to each other row according to a simple rule. One can think of this as choosing some vector c to extend u by, forming the outer product $u \otimes c$, and adding this to the rank decomposition of the matrix.

Unfortunately, slice elimination does not give an efficient algorithm for tensor rank for two reasons. First, unlike in a matrix, the non-eliminated slices can have rank greater than 1 and it will be unclear what the overall rank of the tensor is. Second, even choosing what multiples to use when adding one slice to the others is NP-hard, and this is the property exploited in showing NP-hardness of tensor rank.

2.2 Substitution

While slice elimination does not give an algorithm for tensor rank, it can be used to analyze the rank of some tensors. A rank 1 slice T' of T can be written uniquely as an outer product, so it is natural to assume T' appears in some minimum rank decomposition of T. Indeed, we show this can be assumed in the following lemma. A form of this for polynomials comes from Pan [11] and goes by many names, such as “slice reduction", “layer reduction", and “substitution". We slightly generalize the proof by Håstad [6]. More general versions can be found in papers by Alexeev et al. [1] and Landsberg and Michałek [10]:

▶ **Lemma 4** (Substitution)**.** *Given a d-tensor T of rank r, suppose the 1-slices $T_{1:j}$ for $j \in [k]$ are rank 1 and linearly independent as vectors. Then there is an expansion $T = \sum_{j=1}^{r} v^j \otimes M^j$ with $\mathrm{rk}(M^j) = 1$ and $M^j = T_{1:j}$ for $j \in [k]$.*

Note that the lemma is stated using 1-slices and using coordinates $j \in [k]$ for $T_{1:j}$ for simplicity. It also holds when taking slices in an arbitrary coordinate i and considering an arbitrary list $S \subset [n_i]$ of i-coordinates.

Proof. Suppose $k = 1$ and write $T_{1:1}$ as a linear combination of M^j. We pick one of the M^j and rearrange the equation to write it in terms of $T_{1:1}$ and the other $M^{j'}$. Then we substitute this into the equations for the other slices, thereby eliminating the use of M^j and introducing $T_{1:1}$.

Explicitly, the rank decomposition gives us

$$T_{1:x} = \sum_{j \in [r]} v^j(x) M^j$$

Assume without loss of generality that $v^1(1) \neq 0$. We use this equation to replace all occurrences of M^1 with $T_{1:1}$. First we rearrange the equation for $T_{1:1}$ to get

$$M^1 = \frac{1}{v^1(1)} T_{1:1} - \sum_{j=2}^{r} \frac{v^j(1)}{v^1(1)} M^j$$

Now we can substitute this in the remaining equations and simplify to get

$$\begin{aligned} T_{1:x} &= v^1(x) \left(\frac{1}{v^1(1)} T_{1:1} - \sum_{j=2}^{r} \frac{v^j(1)}{v^1(1)} M^j \right) + \left(\sum_{j=2}^{r} v^j(x) M^j \right) \\ &= \frac{v^1(x)}{v^1(1)} T_{1:1} + \sum_{j=2}^{r} \left(v^j(x) - \frac{v^j(1) v^1(x)}{v^1(1)} \right) M^j \end{aligned}$$

To find an expansion with $M^j = T_{1:j}$ for subsequent j, we simply repeat this procedure. We must be careful that we do not replace an earlier M^j. Fortunately the $T_{1:j}$ for $j \leq k$ are independent, so each one must use some M^j for $j > k$ when it is reached. ◀

We would like to extend this proof to delete rank 1 slices and simplify the tensor. With the same notation as in Lemma 4, write $T = \widetilde{T} + \sum_{h=1}^{k} v^h \otimes M^h$ and note that $\mathrm{rk}(T) = \mathrm{rk}(\widetilde{T}) + k$. We get the following consequence of Lemma 4.

▶ **Corollary 5.** *Given a d-tensor T of rank r, suppose the 1-slices $T_{1:j}$ for $j \in [k]$ are rank 1 and linearly independent as vectors. Then*

$$\mathrm{rk}(T) = k + \min(\mathrm{rk}(T - \sum_{j=1}^{k} v^j \otimes T_{1:j})),$$

where the minimum is taken over the choice of $\{v^j\}_{j \in [k]} \subseteq \mathbb{F}^{n_1}$. Furthermore, in the $T - \sum_{j=1}^{k} v^j \otimes T_{1:j}$ of minimum rank, we can assume slice $1 : j$, for $j \in [k]$, has all zero entries.

In Section 3 we use T' to refer to the minimum rank tensor in Corollary 5 with slices $1 : j$ removed, for $j \in [k]$.

3 Hardness

To establish NP-hardness of approximation we adapt the proof of NP-hardness by Håstad [6] but reduce from bounded occurrence SAT. Håstad started with a 3-SAT instance ϕ with n variables and m clauses and created a $(2+n+2m) \times 3n \times (3n+m)$ tensor T. If ϕ was satisfiable then $\mathrm{rk}(T) = 4n+2m$, and otherwise $\mathrm{rk}(T) > 4n+2m$. We follow the same general approach but by starting with a bounded occurence SAT instance we are able to more tightly relate the rank of the resulting tensor to the minimal number of falsified clauses.

3.1 Reduction

Let us give an overview of our reduction. Though the reduction works for a varying number of literals per clause, here we just reduce from MAX-E2-SAT. Given a SAT formula ϕ with n variables and m clauses, we create a $(1+n+m) \times 2n \times (n+m)$ tensor T with n variable slices and m clause slices. We represent each literal as a vector as follows: $v_{x_i} \in \mathbb{F}^{2n}$ contains a 1 in position $2i-1$ and 0 everywhere else. Vector $v_{\overline{x}_i} \in \mathbb{F}^{2n}$ contains a 1 in positions $2i-1$ and $2i$, and 0 everywhere else. The 3-slices are then defined as follows:

- For $i \leq n$, $T_{3:i}$ represents variable x_i. It has a 1 in positions $(1, 2i-1)$ and $(i+1, 2i)$, and is otherwise 0.
- Suppose the h^{th} clause is $C_h = \ell_1 \vee \ell_2$, where ℓ_i is a literal. Then
  - $(T_{3:(n+h)})_{1:1} = v_{\ell_1}$
  - $(T_{3:(n+h)})_{1:h+n+1} = v_{\ell_1} - v_{\ell_2}$
  - $T_{3:(n+h)}$ is 0 everywhere else

  Here are the slices for the formula $(x \vee y) \wedge (\overline{x} \vee \overline{y})$. We use "." to represent 0 for readability:

$$\begin{bmatrix} 1 & . & . & . \\ . & 1 & . & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} . & . & 1 & . \\ . & . & . & . \\ . & . & . & 1 \\ . & . & . & . \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} 1 & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 1 & . & -1 & . \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} 1 & 1 & . & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 1 & 1 & -1 & -1 \end{bmatrix}$$

This tensor is a subset of the tensor from Håstad's paper [6], with all auxiliary 3-slices and the third copy of the variable columns removed. We use Corollary 5 to reduce the problem of computing the rank of T to that of computing a realization of some matrix M'. The matrix M' consists of:

- For each $i \in [n]$, a row v'_i, representing variable x_i. This row has a 1 in position $2i-1$, a variable a_i in position $2i$, and 0 everywhere else.
- For each $h \in [m]$, a row c'_h, representing clause $C_h = \ell_1 \vee \ell_2$. This row is nonzero only in the columns for variables appearing in the clause. It depends on a variable b_h and equals $(1-b_h)v_{\ell_1} + b_h v_{\ell_i}$.

  For example, for the formula $(x \vee y) \wedge (\overline{x} \vee \overline{y})$ we obtain

$$\begin{bmatrix} 1 & a_1 & . & . \\ . & . & 1 & a_2 \\ 1-b_1 & . & b_1 & . \\ 1-b_2 & 1-b_2 & b_2 & b_2 \end{bmatrix}, \text{ which might yield } \begin{bmatrix} 1 & . & . & . \\ . & . & 1 & 1 \\ 1 & . & . & . \\ . & . & 1 & 1 \end{bmatrix}$$

for a specific assignment to the variables. We use M to represent M' under some assignment to its variables, and call it a **realization** of M'. We use c_h to refer to c'_h and u_i to refer to v'_i under an assignment to the variables of M'. Then we have the following:

▶ **Lemma 6.**

$$\mathrm{rk}(T) = \min(\mathrm{rk}(M)) + n + m$$

where the minimum is taken over all realizations M of M'.

As one might expect, the variables in M' correspond to the choices taken in Corollary 5 of how to subtract slices.

Proof. We perform slice elimination on the 1-slices of T. We think of the third coordinate as being perpendicular to the page, so we can think of 1-slices as cutting horizontally across the page and also through the page. For $i \geq 2$ each 1-slice $T_{1:i}$ is a matrix with a single nonzero row. Only the 3-slice $T_{3:i-1}$ has nonzero values on $T_{1:i}$. Slice $T_{3:i-1}$ is the 3-slice for variable x_{i-1} if $i \leq n$ or clause C_{i-n-1} if $i > n$.

Each of these 1-slices thus has nonzero entries only in positions $T_{1:i}(x, i-1)$ for some x. The 1-slices are thus independent as vectors. We apply Corollary 5 and call the simplified tensor T'. There is a variable a_i in T' obtained from subtracting some multiple of $T_{1:i+1}$ from $T_{1:1}$ for $i \leq n$. There is a variable b_j obtained from subtracting some multiple of $T_{1:n+j+1}$ from $T_{1:1}$. Finally, T' has only size 1 in its 1^{st} coordinate, so we view it as the matrix $T'_{3:1} = M'$ and it has the same rank as T'. ◀

3.2 Bounding the rank

Now we use hardness properties of SAT. We define an *a-good* instance to be a formula ϕ with m clauses where some assignment leaves at most am clauses unsatisfied and an *a-bad* instance ψ to be a formula where every assignment leaves at least am clauses unsatisfied. We use the following theorem from Berman and Karpinski [2]:

▶ **Theorem 7.** *It is NP-hard to distinguish between $(4/792 + \epsilon)$-good and $(5/792 - \epsilon)$-bad instances of E3-OCC-MAX-2SAT, for any $\epsilon > 0$.*

We relate the rank of M to the fraction of clauses that ϕ can satisfy to obtain our main theorem.

▶ **Lemma 8** (Completeness). *If ϕ is a-good then for some realization M of M', we have $\mathrm{rk}(M) \leq n + am$.*

Proof. There is some assignment ρ so $\rho(C)$ is true for the maximum number of clauses $C \in \phi$. Set $a_i = 1 - \rho(x_i)$. Suppose clause $C_h = \ell_1 \vee \ell_2$, where ℓ_1 is a literal of x_i and ℓ_2 is a literal of x_j. If $\rho(\ell_1)$ is true then set $b_h = 0$ so that $c_h = v_{\ell_1} = u_i$. If instead $\rho(\ell_2)$ is true, then set $b_h = 1$ so that $c_h = v_{\ell_2} = u_j$. Thus no satisfied clause contributes to the rank of M. There are only am remaining rows, and hence $\mathrm{rk}(M) \leq n + am$. ◀

We conclude that the variable rows contribute n to the rank and satisfied clause rows contribute nothing. Unsatisfied clause rows contribute to the rank as well, but we have to be careful. In general it is hard to get a precise bound on the contribution of clauses to the rank of M. When many clauses share the same variable, we can only guarantee that the rank increases by some fraction of the number of unsatisfied clauses. However, for the very simple formulas we study, we can establish that Lemma 8 is sharp.

▶ **Lemma 9** (Soundness). *If ϕ is an a-bad instance of E3-OCC-MAX-2SAT then for every realization M of M', we have $\mathrm{rk}(M) \geq n + am$.*

Proof. Fix M, a realization of M' of minimum rank. We build a boolean assignment ρ that falsifies at most $\mathrm{rk}(M) - n$ clauses in ϕ. Say a row **uses** i if its last nonzero entry is in column $2i-1$ or $2i$. Sort the rows of M by the index used.

One row using i is u_i. There are at most 3 more rows using i, which we call r, s, and t, if they exist. Call the corresponding clauses C_r, C_s, and C_t. Let M_i be the submatrix of M consisting of those rows that use values up to i. If $\mathrm{rk}(M_i) - \mathrm{rk}(M_{i-1}) = 1$ then C_r, C_s, and C_t share the same literal of x_i. Call this literal ℓ_i. Then setting $\rho(\ell_i)$ true satisfies all these clauses. Otherwise, set $\rho(\ell_i)$ to satisfy the majority of the clauses C_r, C_s, and C_t.

Because there are only 3 clauses that contain x_i, this leaves at most 1 clause unsatisfied. There are at most $\mathrm{rk}(M) - n$ indices $i \in [n]$ where $\mathrm{rk}(M_i) - \mathrm{rk}(M_{i-1}) > 1$, so we end up with at most $\mathrm{rk}(M) - n$ falsified clauses. Since every assignment to ϕ falsifies at least am clauses, we conclude that $\mathrm{rk}(M) \geq n + am$. ◀

3.3 Inapproximability

Now we get our main theorem.

▶ **Theorem 10** (Theorem 1). *It is NP-hard to approximate 3-tensor rank over any field $\mathbb{F}$ within a factor of $1 + 1/1852 - \delta$, for any $\delta > 0$.*

Proof. We combine Theorem 7, Lemma 8, and Lemma 9. We know $\mathrm{rk}(T) = \min(\mathrm{rk}(M)) + n + m$. If ϕ is a-good then $\min(\mathrm{rk}(M)) \leq n + am$. If ϕ is a-bad then $\min(\mathrm{rk}(M)) \geq n + am$. By Theorem 7 it is NP-hard to distinguish $(4/792 + \epsilon)$-good and $(5/792 - \epsilon)$-bad instances. Hence, it is NP-hard to distinguish whether $\mathrm{rk}(T) \leq 2n + (1 + 4/792 + \epsilon)m$ or $\mathrm{rk}(T) \geq 2n + (1 + 5/792 - \epsilon)m$.

Since $m = \frac{3n}{2}$ in E3-OCC-MAX-2SAT, given $\delta > 0$ we can choose $\epsilon > 0$ so that we have an inapproximability ratio of $1 + 1/1852 - \delta$. ◀

4 Discussion

It seems likely that much better inapproximability results are possible for tensor rank. By counting, we know there are tensors of shape $n \times n \times n$ with rank at least $n^2/3$. Say that a family $T(n)$ of tensors of increasing size n is "explicit" if there is a polynomial time algorithm that takes input n written in unary and prints $T(n)$. Despite the existence of $n \times n \times n$ tensors of quadratic rank, the only "explicit" such tensors have rank at most $O(n)$ [15].

Alexeev et al. [1] claim that "any gap-preserving reduction from NP to tensor rank would automatically yield lower bounds for explicit tensors". While it is not obvious to us how to turn this into a technically precise statement, we sketch the main idea. Suppose we reduce SAT to 3-tensor rank and obtain a superconstant hardness of approximation for tensor rank. If the reduction always outputs a tensor T with largest dimension n, and where the slices in every direction are independent, then the tensor has rank at least n. If, for some $c > 3$, the reduction demonstrates c-hardness of approximation then it must sometimes output tensors of rank at least cn. If this happens on unsatisfiable formulas, we can apply the reduction to get an explicit high rank tensor.

While it is possible that a randomized reduction could avoid this barrier, we are still motivated to find high rank tensors. One approach to finding higher rank explicit 3-tensors or to improving the gap in a reduction is to Kronecker multiply together smaller 3-tensors. It is known [18] that the product is not multiplicative. For the curious reader, let us give a simple counterexample:

▶ **Example 11.** Take a 3-tensor T with slices

$$\begin{bmatrix} 1 & . \\ . & 1 \end{bmatrix} \begin{bmatrix} . & 1 \\ . & . \end{bmatrix}$$

Kronecker multiplying this with itself, we get a 3-tensor with slices

$$\begin{bmatrix} 1 & . & . & . \\ . & 1 & . & . \\ . & . & 1 & . \\ . & . & . & 1 \end{bmatrix} \begin{bmatrix} . & 1 & . & . \\ . & . & . & . \\ . & . & . & 1 \\ . & . & . & . \end{bmatrix} \begin{bmatrix} . & . & 1 & . \\ . & . & . & 1 \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} . & . & . & 1 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix}$$

The former has rank 3, but the latter set can be formed as a linear combination of 8 rank 1 matrices:

$$\begin{bmatrix} 1 & 1 & . & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} . & -1 & . & . \\ . & 1 & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} . & 1 & 1 & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} . & . & . & . \\ . & -1 & . & 1 \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} . & . & . & . \\ . & . & . & -1 \\ . & . & . & -1 \\ . & . & . & . \end{bmatrix}$$

$$\begin{bmatrix} . & . & . & 1 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} . & . & . & . \\ . & . & . & . \\ . & . & 1 & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & 1 \end{bmatrix}$$

T has rank 3, but when Kronecker multiplying with itself it has rank at most 8, not 9. This is true over any field $\mathbb{F}$.

Perhaps it is still true that when k copies of a 3-tensor T of shape $n \times n \times n$ with $\mathrm{rk}(T) > n$ are multiplied together, the rank grows as r^k for some $\mathrm{rk}(T) \geq r > n$. If true, this would immediately give high rank 3-tensors.

References

1. Boris Alexeev, Michael A. Forbes, and Jacob Tsimerman. Tensor rank: Some lower and upper bounds. *CoRR*, abs/1102.0072, 2011. `arXiv:1102.0072`.
2. Piotr Berman and Marek Karpinski. Efficient amplifiers and bounded degree optimization. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(53), 2001. URL: `http://eccc.hpi-web.de/eccc-reports/2001/TR01-053/index.html`.
3. Markus Bläser. On the complexity of the multiplication of matrices of small formats. *J. Complexity*, 19(1):43–60, 2003. `doi:10.1016/S0885-064X(02)00007-9`.
4. Markus Bläser. Explicit tensors. In *Perspectives in Computational Complexity*, pages 117–130. Springer, 2014.
5. Markus Bläser, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Generalized matrix completion and algebraic natural proofs. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:64, 2018. URL: `https://eccc.weizmann.ac.il/report/2018/064`.
6. Johan Håstad. Tensor rank is np-complete. *J. Algorithms*, 11(4):644–654, 1990. `doi:10.1016/0196-6774(90)90014-6`.
7. Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009. `doi:10.1137/07070111X`.
8. Joseph B Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

9 Julian D Laderman. A noncommutative algorithm for multiplying $3\times$ 3 matrices using 23 multiplications. *Bulletin of the American Mathematical Society*, 82(1):126–128, 1976.

10 J. M. Landsberg and Mateusz Michalek. Abelian tensors. *CoRR*, abs/1504.03732, 2015. `arXiv:1504.03732`.

11 Viñtor Yakovlevich Pan. Methods of computing values of polynomials. *Russian Mathematical Surveys*, 21(1):105–136, 1966.

12 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:2, 2010. URL: `http://eccc.hpi-web.de/report/2010/002`.

13 Marcus Schaefer and Daniel Stefankovic. The complexity of tensor rank. *CoRR*, abs/1612.04338, 2016. `arXiv:1612.04338`.

14 Yaroslav Shitov. How hard is the tensor rank? *arXiv preprint arXiv:1611.01559*, 2016.

15 Amir Shpilka. Lower bounds for matrix product. *CoRR*, cs.CC/0201001, 2002. URL: `http://arxiv.org/abs/cs.CC/0201001`.

16 Zhao Song, David P. Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. *CoRR*, abs/1704.08246, 2017. `arXiv:1704.08246`.

17 Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

18 Nengkun Yu, Eric Chitambar, Cheng Guo, and Runyao Duan. Tensor rank of the tripartite state $|w\rangle^{\otimes n}$. *Physical Review A*, 81(1):014301, 2010.