Abstract—Tensors, i.e., multi-linear functions, are a fundamental building block of machine learning algorithms. In order to train on large data-sets, it is common practice to distribute the computation amongst workers. However, stragglers and other faults can severely impact the performance and overall training time. A novel strategy to mitigate these failures is the use of coded computation. We introduce a new metric for analysis called the typical recovery threshold, which focuses on the most likely event and provide a novel construction of distributed coded tensor operations which are optimal with this measure. We show that our general framework encompasses many other computational schemes and metrics as a special case. In particular, we prove that the recovery threshold and the tensor rank can be recovered as a special case of the typical recovery threshold when the probability of noise, i.e., a fault, is equal to zero, thereby providing a noisy generalization of noiseless computation as a serendipitous result. Far from being a purely theoretical construction, these definitions lead us practical random code constructions, i.e., locally random p-adic alloy codes, which are optimal with respects to the measures. We analyze experiments conducted on Amazon EC2 and establish that they are faster and more numerically stable than many other benchmark computation schemes in practice, as is predicted by theory.

I. INTRODUCTION

Machine learning algorithms have become the dominant tool in the broader computing community. A large amount of the success of these algorithms has been due to the availability of large data-sets and more recently the use of hardware optimized to perform multi-linear functions; in particular, GPU's are designed to perform large amounts of matrix multiplications and recently TPU's [1] have been designed to optimize many tensor, i.e., multi-linear operations. This is due to the fact that tensor operations are a fundamental building block for a large number of deep neural network architectures. Furthermore, many machine learning algorithms are themselves a tensor operation; e.g., regression [2], matrix factorization [3], SP networks [5], splines [2], PCA [2], [6], [7], ICA [8], [9], LDA [2], multi-linear PCA [10], [13], SVM [14]–[16], and, more generally, kernel functions/methods [17]–[21]. In statistical inference, the problem of blind source separation [22]–[24] and the related problem of computing the cumulants [25] are related to the tensor rank decomposition since cumulants are tensors that measure the statistical independence between a collection of random variables. Other applications of tensor rank decomposition include data-mining higher order data-sets [26] and interpreting MRI [27], [28]. If the dataset has many datapoints then the overall computation, or job, is distributed as tasks amongst workers, which model a distributed network of computing devices. This solution creates a new problem; stragglers and other faults can severely impact the performance and overall training time.

A. Contribution

This paper offers a novel information-theoretic framework for the analysis of the complexity, numerical stability, and security of distributed fault tolerant tensor computations. Far from being a purely theoretical contribution, we show that our scheme leads us to construct two families of practical code constructions which we call the global random p-adic codes and the locally random p-adic alloy codes; furthermore, we prove that our constructions give better performance than many benchmarks, e.g., the state-of-the-art algorithms in coded distributed computing. As a result, the construction of the locally random p-adic alloy codes gives a procedure for uniformly and effectively converting computationally efficient tensor rank decomposition into fault tolerant distributed coded algorithms; in particular, our algorithm gives an explicit construction that archives the theoretical bound in [29] with (arbitrarily) high probability. This main intuition behind the framework is to consider a probabilistic definition of the recovery threshold that differs from the current one in use; i.e., instead of considering algorithms that satisfy the condition “results from any k out of n workers allows the algorithm to terminate correctly” we consider schemes that satisfy the weaker but more general condition “with probability close to 1, results from the first k out of n workers allows the algorithm to terminate correctly”; i.e., this paper considers the typical fault pattern. All of this culminates in what we call the source channel seperation theorem for coded distributed tensor computation, see Sec. V.

B. Two Motivating Examples

We give the simplest possible examples of a global random p-adic code and a locally random p-adic alloy code. Suppose that we wish to multiply the two matrices $A = (A_1, A_2)^T, B = (B_1, B_2)$. If we have 5 or more workers (i.e., $n \geq 5$), then we can give worker $k$ the coded
the set of all such tensors as 
\[(g_A)_1^k A_1 + (g_A)_2^k A_2, (g_B)_1^k B_1 + (g_B)_2^k B_2\]
where the \(g_i^k\) are i.i.d. according to the distribution in Eq. [3]. We see that the workers will return the values \((g_C)_i^k A_i B_i = (g_A)_i^k (g_B)_i^k A_i B_i\). Lem. [2] gives us that probability of the \((g_C)_i^k\) being an invertible matrix (the \(k\) indexes the rows and the \((i, j)\) indexes the columns) goes arbitrarily close to 1 for a large enough field sizes \(q\) and that this coding scheme works for all fields; in particular, any 4 workers allows us to solve the linear system given by \(G_C\). The coefficients \((g_A)_i^k, (g_B)_i^k, (g_C)_i^k, j\) form a simple global random p-adic code.

In order to illustrate the simplest form of the locally random p-adic alloy code suppose that instead 
\[A = \left(\begin{array}{cccc}
A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4}
\end{array}\right),\]
\[B = \left(\begin{array}{cccc}
B_{1,1} & B_{1,2} & B_{1,3} & B_{1,4} \\
B_{2,1} & B_{2,2} & B_{2,3} & B_{2,4}
\end{array}\right).\]
If we let 
\[A^1 = (A_{1,1} A_{2,1})^T, A^2 = (A_{1,2} A_{2,2})^T, A^3 = (A_{3,1} A_{4,1})^T, A^4 = (A_{2,3} A_{2,4})^T, B^1 = (B_{1,1} B_{1,2}), B^2 = (B_{2,1} B_{2,2}), B^3 = (B_{2,2} B_{2,3}), B^4 = (B_{2,3} B_{2,4})\]
Then Strassen’s algorithm gives us that the matrices 
\[T_1 := (A^1 + A^2)(B^3 + B^4), T_2 := (A^3 + A^4)B^1, T_3 := A^1(B^2 - B^4), T_4 := A^4(B^3 - B^3), T_5 := (A^1 + A^2)B^2, T_6 := (A^3 - A^4)(B^1 + B^2), T_7 := (A^2 - A^4)(B^3 + B^4)\]
are the two matrices \(A' = A^1 + A^4\) and \(B' = B^1 + B^4\) can themselves be further partitioned in two (e.g., \(A' = (A'_1, A'_2)\)) and we can apply the code from the previous example to get coded tensors 
\[(g_A)_1^k A'_1 + (g_A)_2^k A'_2, (g_B)_1^k B'_1 + (g_B)_2^k B'_2\]
likewise the two matrices \(A'' = A^3 + A^4\) and \(B'' = B^3 + B^4\) can be further encoded as 
\[(g_A)_1^k A''_1 + (g_A)_2^k A''_2, (g_B)_1^k B''_1 + (g_B)_2^k B''_2\]
and so on for each \(T_i\). Thus we get 7 codes \((g_A)_1^k, (g_B)_1^k, (g_C)_i^k, j\) for \(t \in [7]\) which allows to solve for the \(T_i\) with high probability from about 28 workers; not all size 28 subsets will solve for the \(T_i\) but nonetheless the typical number of workers needed still ends up being less than EP codes [39] which needs a minimum of 33 workers in this case. Furthermore the master can just drop communication with workers returning useless data so that the master always can receive a maximum of 28 results from the workers.

II. BACKGROUND AND RELATED WORK

If a function \(T\) of the form \(T : V_1 \times \cdots \times V_t \rightarrow V_{t+1}\), where \(V_i \cong \mathbb{F}_q^d\) and \(\mathbb{F}\) is a field, satisfies the equation 
\[T(v_1, \ldots, \alpha v_l + \beta v_1, \ldots, v_l) = \alpha T(v_1, \ldots, v_l) + \beta T(v_1, \ldots, v_l, \ldots, v_l),\]
we say that it is a \(l\)-multi-linear function or that it is a tensor of order \(l\) and we denote 
\[\tau : V_i \rightarrow \mathbb{F}\] as a linear functional if it can

1Some researchers abuse notation and call this number the rank of a tensor, but all researchers agree on the definition of the tensor-rank.

2We abuse notation here since \(V_1 \times \cdots \times V_{t+1} \cong (V_1 \otimes \cdots \otimes V_t)^* \times V_{t+1}\) for finite dimensional \(V_i\).
III. Main Assumptions and Definition of the Model

A. Definition of the Channel

The channel model for our (matrix multiplication) problem is defined as a probability distribution \( p(Y = y | X = x) \) where the random variables \( X \) and \( Y \) take values in the sets \( \mathcal{X} = \{ (x_1, x_2) \mid x_1 \in \mathbb{F}_q P^S, x_2 \in \mathbb{F}_q S^Q \} \). \( Y = \{ y = x_1 \cdot x_2 \mid x_1 \in \mathbb{F}_q P^S, x_2 \in \mathbb{F}_q S^Q \} \cup \{E\} \subset \mathbb{F}_q^P \cup \{E\} \). \( E \) is a placeholder symbol for an erasure. \( y, x_1, x_2 \) are all matrices, and \( \cdot \) is matrix multiplication. We call \( \mathcal{M} = \{ (M_1, M_2) \mid M_1 \in \mathbb{F}_q^{(z_P)(z_S)}, M_2 \in \mathbb{F}_q^{(z_S)(y_Q)} \} \) \( \mathcal{M}' = \{ M_1 \cdot M_2 \mid M_1 \in \mathbb{F}_q^{(z_P)(z_S)}, M_2 \in \mathbb{F}_q^{(z_S)(y_Q)} \} \) the input message set and output message set respectively. A code is a pair of functions \((\mathcal{E}, \mathcal{D})\) where \( \mathcal{E} : \mathcal{M} \rightarrow X^n \) is the encoder and \( \mathcal{D} : Y^n \rightarrow \mathcal{M}' \) is the decoder. The reason for the exponent \( n \) is that we break down a large task \( M_1 M_2 \) into a sequence of \( n \) multiplications \( x_1^1 x_2^1, x_1^2 x_2^2, \ldots, x_1^n x_2^n \) which corresponds to using the channel \( n \) times. A channel must have that \( \sum_y p(y|x) = 1 \), which is to say that given an input something (an output) must occur. For more general tensors \( T \), one defines the input messages as \( \mathcal{M} = \{ (v_1, \ldots, v_l) \mid v_i \in V_l \} \) and the output messages \( \mathcal{M}' = \{ T(v_1, \ldots, v_l) \mid v_i \in V_l \} \cup \{E\} \).

We have a discrete memory-less channel, a channel that has no state and that the output \( Y \) depends only on the input \( X \) so that \( p(y|x) \) is conditionally independent of all other variables. In particular we have that \( p(Y^n = y^n | X^n = x^n) = \prod_i p(Y_i = y_i | x_i = x_i^j) \) This models the stronger assumption that each worker’s failure probability is i.i.d; which is a weak assumption in a distributed environment with simple 1-round master to worker communication.

1) Different Types of codes: Random codes are probability distributions over all possible codes \( \mathcal{C} = (\mathcal{E}, \mathcal{D}) \). The deterministic case is a special case of the non-deterministic case. Tensor codes for matrix multiplication are codes of the form \( \mathcal{E} = G_A, G_B \) and \( \mathcal{D} = \text{solving the equation } G_C(A \cdot B)_{\text{flat}} = M^n \) where \( G_A \ast G_B = G_C, \ A, B \) are block matrices, and \( \text{flat} \) converts a matrix to a vector; i.e.,

\[
(gc)_{i,j} = (g_A)_{i,j} (g_B)_{j,i},
\]

where \( k \) corresponds to the \( k \)th worker. For higher order tensors the code is defined as

\[
(gc)_{i_1,\ldots,i_l} = (g_A)_{i_1,\ldots,i_l} (g_A)_{l,i},
\]

It is important to note that the operation \( \ast \) is not matrix multiplication but instead defined by Eq. [7] and Eq. [8]. The justification for the definition in Eq. [8] is best understood by the discussion in Sec. [4] we know that \( T((g_A)_{i_1,\ldots,i_l} v_i, \ldots, (g_A)_{l,i} v_l) = (gc)_{i_1,\ldots,i_l} T(v_1,\ldots,v_l) \) by multi-linearity and thus, by giving the workers some coded \( \tilde{v} \), on some simpler \( \tilde{T} \), we can recreate the desired \( T(v_1,\ldots,v_l) \) from a sufficient number of \( \tilde{T}(\tilde{v}) \) (once again by multi-linearity).

B. The Capacity of Our Channel

There are two definitions of capacity: the information theoretic and operational. Due to the open status of the complexity of matrix multiplication for our definition it unknown whether the definitions are equal as in the classical Shannon theory case; however, in the case of “2D” distributed multiplication (see Lem[1] they two are equal. The information theoretic definition is given by \( C = \max_{p(x,y)} I(X,Y) \) where \( I(X,Y) \) is the mutual information of the variables \( X,Y \) and \( \mathcal{D}(\mathcal{X}) \) is the set of all distributions on \( X \). \( I(X,Y) \) is defined as \( H(Y) - H(Y|X) \) where \( H(Y) = \sum_y p(y) \log \frac{1}{p(y)} \) is the entropy and \( H(Y|X) = \sum_{x,y} p(x) H(Y|X = x) \) is the conditional entropy.

Lemma 1. The capacity of the channel in Sec. III-A is equal to \( C = (1 - p_f) H(Y \setminus \{E\}) = (1 - p_f) H(p_1, \ldots, p_{|Y|-1}) \).

Typically in most channel models we have that the input message set, \( \mathcal{M} \) (the domain of the encoder \( f \)) is equal to the output message set, \( \mathcal{M}' \) (the range of the decoder \( g \)) Not only do they not necessarily equal one another but they can have different sizes If we had that \( \mathcal{M} \subseteq \mathcal{M}' \) or \( |\mathcal{M}| \leq |\mathcal{M}'| \) then perhaps it would be trivial to fix but we commonly have that \( |\mathcal{M}| \geq |\mathcal{M}'| \). Therefore care must be taken with the calculations and the definitions. If we let the code-book be defined as \( C = (\mathcal{E}, \mathcal{D}) \) then the probability of error for the channel \( p(Y|X) \) is usually given as the distribution \( p_C(i) = p(D(Y^n) \neq i | X^n = \mathcal{E}(i)) \). For this definition the fix is simple, we have that the correct definition should be \( p_C(M_1, M_2) = p(D(Y^n) \neq M_1 M_2 | X^n = \mathcal{E}(M_1, M_2)) \). A code \( C \) that has \( M = |\mathcal{M}| \) input messages, code-words \( X^n \) of length \( n \), and probability of error \( \epsilon \) is said to be a \( (M, n, \epsilon) \)-code. Usually the \( \epsilon \) is dropped and we call it a \( (M, n) \)-code. The (classical) rate of a \((M,n)\)-code is usually given as \( R = \frac{\log |\mathcal{M}|}{n} \); however, we give the definition of the (computational) rate of a \((M,n)\)-code as \( R = \frac{\log (|\mathcal{M}'|\cdot \mathcal{E})}{n} \) where \( \mathcal{E} \) is the set of output sequences with an erasure.

The new definition of the rate and the old definition satisfy the following relationship \( R = \frac{\log (|\mathcal{M}'|\cdot \mathcal{E})}{n} \). The main reason for this change is that we will soon prove that we can perform coded matrix multiplication at a rate sharply bounded as \( R \leq C \) if we were to instead use the classical definitions of \( \tilde{R}, \tilde{C} \) we would have nonsensical scenarios where \( R > \tilde{C} \).

C. Typical Recovery Threshold

The typical sets, \( \mathcal{A}_r \), are defined as the set of sequences \( (x_1, \ldots, x_n) \) such that \( H(X) - \epsilon \leq -\frac{1}{n} \log p(x_1, \ldots, x_n) \leq H(X) + \epsilon \) This is equivalent to \( 2^{-n(H(X)+\epsilon)} \leq p(x_1, \ldots, x_n) \leq 2^{-n(H(X)-\epsilon)} \). The typical recovery threshold, \( R(x,y,z, p_f, \epsilon, C) \), is the number of workers needed for the code \( C \) to have less than \( \epsilon \) probability of error for a partition of type \((x,y,z)\) if each worker has a \( p_f \) probability of fault. The typical recovery threshold and the rate satisfy the following relationship \( R = \frac{\log (|\mathcal{M}'|\cdot \mathcal{E})}{n} \). This is because for a \((M,n)\)-code \( C \) we have that the typical recovery threshold
is equal to \( R = n \) because \( n \) was defined as the number of workers needed. In order to generalize to arbitrary fields, we measure the rate in units that simplifies the relationship between the rate and the recovery threshold as \( R = \frac{m}{\tau} \), i.e., our unit of information equals \( \log(|\mathcal{F}| - 1) \).

IV. GLOBAL RANDOM P-ADIC ALLOY CODES

If we let the uniformly random p-adic 2-product distribution be defined as

\[
p^*(g_j^z) = \begin{cases} 1 - \sqrt{\frac{-q}{q}} & \text{for } z = 0 \\ \frac{1}{\sqrt{q(q-1)}} & \text{for } z \neq 0 \\ \end{cases} (3)
\]

and the uniformly random p-adic tensor distribution, or the uniformly random p-adic 1-product tensor distribution, be defined as

\[
p^*(g_j^z) = \begin{cases} 1 - \sqrt{\frac{-q}{q}} & \text{for } z = 0 \\ \frac{1}{\sqrt{q(q-1)}} & \text{for } z \neq 0 \\ \end{cases} (4)
\]

then we can define our random p-adic codes as

\[
A_k = \sum_{i \in [x]} (g_A)^k_i A_{i}, \quad B_k = \sum_{i \in [y]} (g_B)^k_i B_{i},
\]

for matrix multiplication and more generally as

\[
(A_1, ..., A_l) = \left( \sum_{i \in [x]} (g_A)^k_i A_{1,i}, ..., \sum_{i \in [x]} (g_A)^k_i A_{l,i} \right).
\]

Lemma 2 (Existence of Random Tensor codes for Matrix Multiplication). There exists a probability distribution on \( G_A, G_B \) such that \( G_C = G_A \ast G_B \) has its coefficients uniformly i.i.d.: in particular, the probability of success for the p-adic scheme is \( \prod_{i=1}^{l}[1-q^{-i}] > (1 - q^{-1})^k \), where \( k \) is the maximum rank of \( G_C \).

Proof: We place the distribution \( p^* \) from Eq. 3 on \( G_A, G_B \) then we get that if \( p((g_c)^k_i = z) = \sum_{xy = z} \frac{1}{q} p((g_A)^k_i = x) \) then \( p((g_B)^k_i = y) \). Because \( (g_c)^k_i = (g_A)^k_i (g_B)^k_i \) and the distributions \( p_A, p_B \) are independent, since \( p_A, p_B \) are also identically distributed we can set \( u = p^*(g_j^z = 0) = 1 - \sqrt{\frac{-q}{q}} \) and \( v = p^*(g_j^z) = \frac{1}{\sqrt{q(q-1)}} \). Therefore we get that

\[
p^*((g_c)^k_{i,j} = z) = \begin{cases} u^2 + 2(q-1)uv & \text{for } z = 0 \\ (q-1)v^2 & \text{for } z \neq 0 \\ \end{cases}
\]

This is because there is 1 way to get \( x, y = 0 \), there are \( 2(q-1) \) ways to multiply some \( x, y \neq 0 \) with a zero to get a zero, and there are \( q-1 \) ways to get \( x, y \neq 0 \) to equal \( z \neq 0 \). These three cases correspond to the probabilities of \( u^2, uv \), and \( v^2 \) respectively. Therefore the probability of a 0 is equal to \( p^*((g_c)^k_{i,j} = 0) = u^2 + 2(q-1)uv = \left(1 - \sqrt{\frac{-1}{q}}\right)^2 + 2(q-1) \left(1 - \sqrt{\frac{-1}{q}}\right) \left(\frac{1}{\sqrt{q(q-1)}}\right) = \frac{1}{q} \).

Similarly we have that \( p^*((g_c)^k_{i,j} = z) = (q-1)v^2 = (q-1) \left(\frac{1}{\sqrt{q(q-1)}}\right)^2 = \frac{1}{q} \), for a non zero \( z \).

Because \( p^*((g_c)^k_{i,j} = z) \) is a function of the values \( p^*((g_A)^k_i = 0), ..., p^*((g_A)^k_i = q - 1) \) and \( p^*((g_B)^k_i = 0), ..., p^*((g_B)^k_i = q - 1) \) we have that \( k \neq k' \) implies that \( p^*((g_c)^k_{i,j} = z) \) and \( p^*((g_c)^k_{j,i} = z) \) are independent. Therefore we have that any two rows of \( G_C \) are independent. Similarly if either \( i \neq i' \) or \( j \neq j' \) we have that \( p^*((g_c)^k_{i,j} = z) \) and \( p^*((g_c)^k_{j,i} = z) \) are independent. Therefore we have independence of \( G_C \) across columns.

Lemma 3 (Noisy Channel Coding Theorem for (2-D) Distributed Matrix Multiplication). If we have \( R = C - \epsilon \) and \( z = 1 \) then random alloy codes achieve the rate \( R \) for large enough \( x, y \). Conversely if \( R > C \) the probability of error cannot be made arbitrarily small for any tensor code.

Proof: (Achievability) Fix an \( \epsilon \) and generate product codes \( C = (G_C, G_A, G_B) \) randomly and let \( E_1 = P(G_C \text{ fails}), E_2 = P(\# \text{erasures} > r) \), then we have that the probability of error is equal to \( P(E_1) \) where \( E_1 = P(E_1 \lor E_2) \leq P(E_1) + P(E_2) \). If \( S_G^{xy} \) is the random variable corresponding to choosing a random \( xy \times xy \) sub-matrix of \( G_C \) then \( S_G^{xy} \) is also uniformly randomly distributed and in particular \( P(E_1) = P(G_C \text{ fails}) \leq P(\text{rank}(S_G^{xy}) < xy) \). This is because \( P(\text{rank}(S_G^{xy}) < xy) \) also computes the probability that after fixing some subset \( W = \{w_1, ..., w_{xy}\} \) of the workers that the rows of \( G_C \) corresponding to \( W \) will be a full rank matrix.

Therefore we have that \( P(E_1) \leq \prod_{i=1}^{l}[1-q^{-i}] \). If we let \( p_c^C(M_1, M_2) = P(E_2 = M | G_C) = P(E_2 = G_C) \), then we have that \( P(E_2) \leq \sum_{G_C,M_1,M_2 \in M} p_c^C(M_1, M_2) p(M_1, M_2, G_C) \) for any fixed \( M_1, M_2, \) and \( G_C \) we have that \( p_c^C(M_1, M_2) = P(Y > r \text{ errors}) \). For any fixed input \( X \) we have that either \( Y = x_1 x_2 \) or \( Y = W \) with probabilities \( 1 - p_f \) and \( p_f \). Therefore for any fixed \( M_1, M_2, \) and \( C = (G_A, G_B, G_C) \) we have that \( Y \) depends on only the fixed input \( X \) and \( Y = E(M_1, M_2, M_A, M_B) \). In particular \( M_1, M_2, \) and \( G_C \) being fixed implies \( Y \) is a Bernoulli process and the proof follows from classical the classical version.

(Converse) Suppose for a contradiction that the capacity bound were violated by some product code \( C = (E, D) \), we derive a contradiction to the original Shannon theorem. Recall that the original definition of an error was \( p_c^E(M_1, M_2) = P(D(Y > r) \neq M_1 M_2 | X = E(M_1, M_2)) \). With this definition in mind we define a new message set \( \mathcal{N} \) as follows

\[
\mathcal{N} = \{ S \subset M | (M_1, M_2) \in S \land (M_3, M_4) \in S \} \implies M_1 M_2 = M_3 M_4, \text{ clearly defines a set of equivalence classes } [M_1, M_2] \sim \text{ corresponding to the equivalence relation } (M_1, M_2) \sim (M_3, M_4) \text{ } \overset{def}{=} \text{ } M_1 M_2 = M_3 M_4. \text{ Further more it is clear that } |\mathcal{N}| = |M| \setminus \mathcal{E}, \text{ i.e., that has as many elements as the output set. For each equivalence class } [M_1, M_2], \text{ pick a distinct representative (this can be done because } [M_1, M_2] \sim \text{ and } \mathcal{N} \text{ are finite). Suppose that we have random variable } Z \text{ that takes values in the set } \mathcal{Z} \text{ and that } |\mathcal{Z}| = |\mathcal{N}|. \text{ Since } |\mathcal{Z}| = |\mathcal{N}| \text{ there is a function } h : \mathcal{Z} \rightarrow \mathcal{N} \text{ that is 1-1 and unto (and } \mathcal{Z}, \mathcal{N} < \infty \text{ imply that } h \text{ is computable) Further let } j : \mathcal{N} \rightarrow \mathcal{M} \text{ be the function sends } [M_1, M_2] \sim \text{ to } (M_1, M_2). \text{ Then the}
code defined as $C' = (E_{j}h, h^{-1}O_{-1}D)$ violates the classic Shannon theorem. This is because we defined the probability of error for a product code to be $p_{C}(M_{1}, M_{2}) = p(D(Y) \neq M_{1}M_{2} | X = \epsilon(M_{1}, M_{2}))$. Therefore if $h(z) = [M_{1}, M_{2}]$ then we have that the classical definition of an error is equal to $p_{C}(z) = p((h^{-1}j^{-1}D)(Y) \neq z | X = (E_{j}h)(z)) = p((j^{-1}D)(Y) \neq [M_{1}, M_{2}] | X = (E_{j})[M_{1}, M_{2}] \sim) = p(D(Y) \neq M_{1}M_{2} | X = \epsilon(M_{1}, M_{2})) = p_{C}(M_{1}, M_{2})$.

Theorem 1 (Source Channel Separation Theorem for 3-D Distributed Matrix Multiplication). If we have $R = C - \epsilon$ then local random p-adic code allows achieve the rate R for large enough x, y, z. Conversely if $R > C$ the probability of error cannot be made arbitrarily small for any tensor code.

**Proof Sketch:** Let $T = T_{1} + \ldots + T_{r}$ be an optimal tensor rank decomposition for $(x, y, z)$ then we can apply the code in Eq. 5 to the $T_{i}$ to get r optimal $(\lambda_{1}, \lambda_{2}, 1)$ codes. Then this creates a block code for the larger $T$ via standard argements by the method of strong types, i.e., Hoeffding’s inequality, as found in [42]. Taking appropriate limits for the $\lambda_{1}, \lambda_{2}$ as $x, y, z \to \infty$ using the previous claim completes the proof.

V. LOCALLY RANDOM P-ADIC ALLOY CODES

The general local p-adic alloy code is given by Alg. 1. It is easier to understand it in the case where $T$ is the block matrix multiplication from Sec. [1B] where the $T$ that the workers perform is simply a smaller matrix multiplication. In this case we have that $E_{1}(A) = A^{1} + A^{1}$ and $E_{2}(B) = B^{1} + B^{1}$ and so on for $T_{2}, \ldots, T_{7}$. The $D_{i}$ are simply multiplication by $\pm 1$. In general, through a simple recursive arguement, all fast matrix multiplications can be turned into such tensor decompositions.

**Algorithm 1 Locally_Random_P-adic_Aloy_Codes**

**Input:** tensor $T$, decomposition $T_{1}, \ldots, T_{r}$, and vectors $A_{1}, \ldots, A_{l}$ where $T = T_{1} + \ldots + T_{r}$, $T_{i}(A) = D_{i}(T_{i}(E_{1}(A_{1}), \ldots, E_{l}(A_{l})))$, the workers perform $T_{i}$, and the $E_{j}, D_{i}$ are low complexity tensors. workers $w_{k}^{t}$ indexed by $(t, k)$ where $t \in [r], k \in [n/r]$.

for $r \leq r$ do

Master node generates $g^{t} := (g_{A_{j}}^{t})_{i}^{k}$ according to Eq. 4

Master computes $A_{k,j}^{t} = \sum_{e \in [e]}(g_{A_{j}}^{t})_{e}^{k}E_{j}^{t}(A_{j,i})$

Send worker $(t, k)$ the tasks $A_{k,1}^{t}, \ldots, A_{k,t}$

end for

Each worker $(t, k)$ computes $T_{i}(A_{k,1}^{t}, \ldots, A_{k,t})$

while Master has not received $R^{t}$ tasks for each $t \in [r]$ do

Master receives $T_{i}(A_{k}^{t})$ from next worker $(k, t)$

end while

for $t \leq r$ do

Decode the $T_{i}(A_{k}^{t})$ by inverting $C_{i}$

end for

return $(D_{1}T_{1} + \ldots + D_{r}T_{r})(A_{1}, \ldots, A_{l})$

VI. EXPERIMENTAL RESULTS

The experiments were performed on Amazon EC2 with a r5dn.xlarge for the master and c5n.large for the workers. We compared our algorithm with Entangled Polynomial Codes and Khatari Rao Codes for the numerical stability experiments and simply with EP for the 3-D runtime experiments. For the runtime experiments we used (uniform and normal) randomly generated matrices of size 9600x9600 for the runtime experiments and 1000 times the partition parameters $(x, y, z)$ for the numerical stability/error experiments; i.e., we performed experiments on size $6000 \times 2000$ and $4000 \times 3000$ for the error measurements. For the 3-D runtime experiments we used networks of 22, 25, 36, 41, 50, 57, 71, 78, and 81 machines (including the master). For the 2-D numerical stability experiments we used networks of size 7, 10, 13, 17, 21, 26 machines. For a fair comparison with the EP codes we ran two types of experiments: one with the same number of workers and one with the typical recovery threshold plus seven workers.

VII. CONCLUSION

This paper proposed a novel frame-work for uniformly and effectively converting computationally efficient tensor rank decomposition into fault tolerant distributed coded algorithms. Our approach allows for powerful, time-tested channel coding techniques from information theory to be combined with the algebraic techniques from symbolic computation which introduces novel and interesting problems to the information theory community that provides a new bridge between the
information theoretic communication complexity and algebraic computational complexity by allowing time-tested channel coding techniques from theory to be combined with the algebraic techniques from symbolic computation.

References

[1] N. P. Jouppi, C. Young, N. Patil et al., “In-datacenter performance analysis of a tensor processing unit,” 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), pp. 1–12, 2017.
[2] T. J. Hastie, R. Tibshirani, and J. H. Friedman, “The elements of statistical learning: Data mining, inference, and prediction, 2nd edition,” in Springer Series in Statistics, 2009.
[3] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” Nature, vol. 401, pp. 788–791, 1999.
[4] ———, “Algorithms for non-negative matrix factorization,” in NIPS, 2000.
[5] H. Poon and P. M. Domingos, “Sum-product networks: A new deep architecture,” 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 689–690, 2011.
[6] H. Hotelling, “Analysis of a complex of statistical variables into principal components,” 1933.
[7] B. Schölkopf, A. J. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” Neural Computation, vol. 10, pp. 1299–1319, 1998.
[8] P. Comon, “Independent component analysis, a new concept?” Signal Process., vol. 36, pp. 287–314, 1994.
[9] S. J. Roberts and R. M. Everson, “Independent component analysis: Principles and practice,” 2001.
[10] P. M. Kroonenberg and J. de Leeuw, “Principal component analysis of three-mode data by means of alternating least squares algorithms,” Psychometrika, vol. 45, pp. 69–97, 1980.
[11] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, “On the best rank-1 and rank-(r1 , r2, ... , rn) approximation of higher-order tensors,” SIAM J. Matrix Anal. Appl., vol. 21, pp. 1324–1342, 2000.
[12] ———, “A multilinear singular value decomposition,” SIAM J. Matrix Anal. Appl., vol. 21, pp. 1253–1278, 2000.
[13] M. A. O. Vasilescu, “Multilinear independent component analysis,” 2004.
[14] B. E. Boser, I. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in COLT ’92, 1992.
[15] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, pp. 273–297, 2004.
[16] N. Cristianini and J. Shawe-Taylor, “An introduction to support vector machines and other kernel-based learning methods,” 2000.
[17] M. A. Aizerman, “Theoretical foundations of the potential function method in pattern recognition,” 1964.
[18] D. Haussler, “Convolution kernels on discrete structures,” 1999.
[19] C. Watkins, “Kernels from matching operations,” 1999.
[20] ———, “Dynamic alignment kernels,” 1999.
[21] J. Shawe-Taylor and N. Cristianini, “Kernel methods for pattern analysis,” 2004.
[22] P. Comon and C. Jutten, “Handbook of blind source separation: Independent component analysis and applications,” 2010.
[23] N. D. Sidiropoulos, G. B. Giannakis, and R. Bro, “Blind parafac receivers for ds-cdma systems,” IEEE Trans. Signal Process., vol. 48, pp. 810–823, 2000.
[24] L. D. Lathauwer and A. de Baynast, “Blind deconvolution of ds-cdma signals by means of decomposition in rank-S(1,1,1,...,1) terms,” IEEE Transactions on Signal Processing, vol. 56, pp. 1562–1571, 2008.
[25] T. S. Rao and P. McCullagh, “Tensor methods in statistics,” 1989.
[26] X. Li, S. Lin, S. Yan et al., “Discriminant locally linear embedding with high-order tensor data,” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 38, pp. 342–352, 2008.
[27] T. Schultz and H.-P. Seidel, “Estimating crossing fibers: A tensor decomposition approach,” IEEE Transactions on Visualization and Computer Graphics, vol. 14, 2008.
[28] F. Calamante, M. Mrup, and L. K. Hansen, “Defining a local arterial input function for perfusion mri using independent component analysis,” Magnetic resonance in medicine, vol. 52 4, pp. 789–97, 2004.
[29] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding,” 2018 IEEE International Symposium on Information Theory (ISIT), pp. 2022–2026, 2018.