GTT: Guiding the Tensor Train Decomposition

Mao-Lin Li[1], K Selçuk Candan[1], and Maria Luisa Sapino[2]

[1] Arizona State University, Tempe AZ, USA
{maolinli,candan}@asu.edu
[2] University of Turin, Turin, Italy
mlsapino@di.unito.it

Abstract. The demand for searching, querying multimedia data such as image, video and audio is omnipresent, how to effectively access data for various applications is a critical task. Nevertheless, these data usually are encoded as *multi-dimensional* arrays, or *Tensor*, and traditional data mining techniques might be limited due to the *curse of dimensionality*. Tensor decomposition is proposed to alleviate this issue, commonly used tensor decomposition algorithms include CP-decomposition (which seeks a diagonal core) and Tucker-decomposition (which seeks a dense core). Naturally, Tucker maintains more information, but due to the denseness of the core, it also is subject to exponential memory growth with the number of tensor modes. Tensor train (TT) decomposition addresses this problem by seeking a sequence of three-mode cores: but unfortunately, currently, there are no guidelines to select the decomposition sequence. In this paper, we propose a GTT method for guiding the tensor train in selecting the decomposition sequence. GTT leverages the data characteristics (including number of modes, length of the individual modes, density, distribution of mutual information, and distribution of entropy) as well as the target decomposition rank to pick a decomposition order that will preserve information. Experiments with various data sets demonstrate that GTT effectively guides the TT-decomposition process towards decomposition sequences that better preserve accuracy.

Keywords: Low-rank embedding, Tensor train decomposition

1 Introduction

Tensors are commonly used to represent multi-dimensional sets and tensor decomposition operations, such as CP [5, 10] and Tucker [24] form the basis of many dimensionality reduction techniques for multi-modal data sets to support similarity search and retrieval [4, 11]. In the Tucker-decomposition, for example, given a tensor with d modes, each entry in the resulting $r_1 \times r_2 \times \ldots \times r_d$ dense core encodes the strength of the d-way relationship among the groups consisting of elements of the individual modes.

Tucker decomposition has been shown to be highly effective in many applications [4, 11, 18, 25], but due to the denseness of the core, it also is subject to exponential memory growth with the number of tensor modes. The tensor

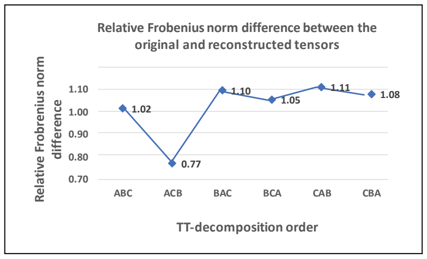


Fig. 1. Effect of the decomposition order on the accuracy for a 3-mode tensor from the Wisconsin Diagnostic Breast Cancer data [9]: $ID(mode_A)$, $Diagnosis(mode_B)$ and $Radius(mode_C)$. See Section 6 for more details

train (TT) decomposition addresses this problem, by seeking a sequence of 3-mode cores [23]: while, collectively, this sequence (or "train") of cores capture the high-modal information, they require fewer resources. Consequently, the TT-decomposition has been used in various applications of similarity search and retrieval, including deep learning [6, 21], crowdsourcing [16], and recommendation systems [22].

1.1 Impact of the Decomposition Order

One critical challenge with the TT-decomposition, however, is the fact that finding an optimal TT representation is non-trivial [27]. Figure 1 illustrates this issue: given a 3-mode ($mode_A$: *ID*, $mode_B$: *Diagnosis* and $mode_C$: *Radius*) tensor from the Wisconsin Diagnostic Breast Cancer data set in UCI Machine Learning Repository [9]; the figure compares the relative Frobrenius norm difference (ratio of the norm of the difference tensor to the norm of the original tensor) between the input tensor and the reconstructed tensor for different TT-decomposition orders. As the figure shows, the ordering of the TT-decomposition has a significant impact on the ability of the final representation in preserving the original information: in this case, the order ACB is $(0.77 - 1.02)/1.02 = 24.5\%$ better than the closest alternative.

1.2 Our Contributions

In this paper, we propose a novel approach for *guiding the tensor train* (GTT) in selecting the mode sequence for tensor train decompositions:

- we identify significant relationships among various data characteristics and the accuracies of different tensor train decomposition orders;
- we propose four order selection strategies, (a) *aggregate mutual information (AMI)*, (b) *path mutual information (PMI)*, (c) *inverse entropy (IE)*, and (d) *number of parameters (NP)*, for tensor train decomposition; and
- we show that good tensor train orders can be selected through a *hybrid (HYB)* strategy that takes into account multiple characteristics of the data.

Experiments reported in Section 6 show that the proposed HYB strategy provides an effective order selection strategy, without any additional decomposition time overhead.

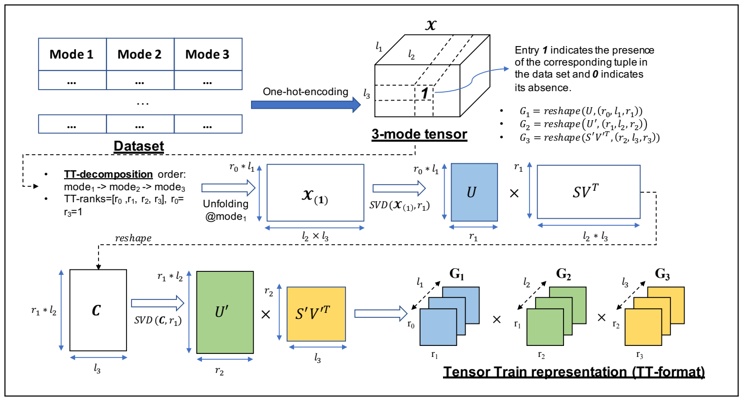

Mode 1
Mode 2
Mode 3
...
Dataset
One-hot-encoding
$\mathcal{X}$
l_1
l_2
l_3
1
3-mode tensor
Entry *1* indicates the presence of the corresponding tuple in the data set and *0* indicates its absence.
$G_1 = reshape(U, (r_0, l_1, r_1))$
$G_2 = reshape(U', (r_1, l_2, r_2))$
$G_3 = reshape(S'V'^T, (r_2, l_3, r_3))$
TT-decomposition order: mode$_1$ -> mode$_2$ -> mode$_3$
TT-ranks=[r_0, r_1, r_2, r_3], r_0= r_3=1
Unfolding @mode$_1$
$r_0 * l_1$
$\mathcal{X}_{(1)}$
$l_2 \times l_3$
$SVD(\mathcal{X}_{(1)}, r_1)$
U
r_1
SV^T
$l_2 * l_3$
reshape
$r_1 * l_2$
C
$SVD(C, r_1)$
U'
r_2
$S'V'^T$
$\mathbf{G_1}$
$\mathbf{G_2}$
$\mathbf{G_3}$
r_0
r_1
r_2
r_3
Tensor Train representation (TT-format)

Fig. 2. TT-decomposition for converting a 3-mode tensor $\mathcal{X}^{l_1 \times l_2 \times l_3}$

2 Related Work

Tensor decomposition has been shown to be effective in multi-aspect data analysis and similarity search by capturing high-order structure in high-dimensional data [4, 11, 18]. However, a major challenge is its high computational complexity and large memory overhead [12, 14, 15]. Tensor-train decomposition [23] provides a memory-saving representation called *TT-format*, with linear space complexity (see Figure 2). TNrSVD [2] adapts the randomized SVD to implement TT-decomposition, and FastTT [19] computes the TT-decomposition of a sparse tensor by its sparsity. However, as discussed in the introduction, TT-decomposition involves strictly sequential multi-linear products over latent cores and this makes it difficult to search for best TT representation for a given tensor. [20] and [27] extended TT-decomposition by adding auxiliary variables to obtain an alternative data structure, Tensor Ring (*TR*), which provides *circular dimensional permutation invariance* – the sequence can be shifted circularly without changing the result [1], however, it does not eliminate the need to pick a (*circularly-arranged*) permutation of modes.

In this paper, we focus on the problem of selecting a tensor decomposition order, which is superficially related to feature ordering and feature selection [3, 17, 26], which search for the most relevant attributes of the data set (for a given application) to reduce the dimensionality. *Entropy*, for example, tends to be low for data that contain tight clusters [7, 8]. Various other data characteristics, such as *variance*, *mutual information*, have been used for selecting the order of decisions in supervised machine learning, such as decision trees [4].

3 Preliminaries

Table 1 summarizes the key notations. Intuitively, the tensor model maps a schema with d attributes to a d-modal array (where each potential tuple is a tensor cell). TT-decomposition [23] is obtained by applying a sequence of singular value decompositions (SVD) to approximate the original tensor: given

Table 1. Notations used in the paper

	Description
$\mathcal{X}$	A tensor
$\rho_{\mathcal{X}}$	Density of tensor $\mathcal{X}$
$\mathcal{X}_{(i)}$	A mode-i unfolding matrix of a tensor $\mathcal{X}$
l_i	Length of mode i in a tensor
m_i	The mode i in a tensor
r_i	TT-rank of mode i
r_{max}	Given maximum TT-rank
X	A discrete random variable with possible values $\{x_1, \ldots, x_n\}$
U	Left factor matrix
S	Singular matrix
V	Right factor matrix
G_i	3-mode core for mode i of TT-decomposition
H_i	Shannon entropy of random variable for mode i
$H_{i\|j}$	Conditional entropy for mode i given mode j
$H_{(i,j)}$	Averaged conditional entropy for $H_{i\|j}$ and $H_{j\|i}$
$MI_{(i,j)}$	Mutual information between mode i and mode j

Algorithm 1 TT-SVD (adapted from [23])

```
Input:
A $d$-mode tensor $\mathcal{X} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_d}$; a target tt-rank, $r$; a permutation, $\Pi$
Output:
TT-format with TT-cores $G_1, G_2, \ldots G_d$.
• $numel(C)$ : number of elements in $C$.
• $reshape(A, [d_1, \ldots, d_k])$ : reshape a tensor $A$ into shape $d_1 \times d_2 \times \cdots \times d_k$ .
• $min(a, b)$ : return $a$ if $a < b$, else return $b$.
1: procedure TT-SVD($\mathcal{X}, r, \Pi$)
2:     Initialize $r_0 = r_d = 1,\ C = \mathcal{X}$.
3:     for $k \leftarrow 1$ to $d - 1$ do
4:         $C \leftarrow reshape(C, [r_{k-1} \times l_{\pi_k}, \frac{numel(C)}{r_{k-1} \times l_{\pi_k}}])$.
5:         $U, S, V = SVD(C, r_k = min(r_{max}, l_{\pi_k}))$.
6:         $G_k \leftarrow reshape(U, [r_{k-1}, l_{\pi_k}, r_k])$.
7:         $C \leftarrow SV^T$.
8:     end for
9:     $G_d \leftarrow C$.
10:    return TT-format with TT-cores $G_1, \ldots, G_d$.
11: end procedure
```

(i) an input tensor, $\mathcal{X} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_d}$, (ii) a permutation, $\Pi = \langle \pi_1, \pi_2, \ldots, \pi_d \rangle$, of modes, and (iii) a sequence of decomposition ranks, $\langle r_0, r_1, r_2, \ldots, r_d \rangle$, where $r_0 = r_d = 1$, the tensor train decomposition approximates the input tensor, $\mathcal{X}$, with a sequence of tensor cores $G_k \in \mathbb{R}^{r_{k-1} \times l_{\pi_k} \times r_k}, k = 1 \ldots d$, where $\mathcal{X} \approx \hat{\mathcal{X}}_{\Pi} = G_1 \cdot G_2 \cdot \cdots \cdot G_d$. In this paper, without loss of generality, we will assume that all ranks (except $r_0 = r_d = 1$) have the same value, r. Note that, while there are several non-parametric decomposition techniques, such as [13] which can learn also the appropriate rank, this is outside of the scope of this paper – most tensor decomposition (in fact most latent semantic search) literature takes the number of latent-semantics as input. Algorithm 1 presents the pseudocode and Figure 2 visualizes the TT-SVD process for a 3-mode tensor $\mathcal{X} \in \mathbb{R}^{l_1 \times l_2 \times l_3}$.

Accuracy. To evaluate the accuracy, we use the Frobenius norm of the difference between mode-i unfolding $\mathcal{X}_{(i)}$, of the original tensor and mode-i unfolding $\hat{\mathcal{X}}_{\Pi_{(i)}}$ of the reconstructed tensor, $\hat{\mathcal{X}}_{\Pi}$: $Error(\hat{\mathcal{X}}_{\Pi}, \mathcal{X}) = \|\mathcal{X}_{(i)} - \hat{\mathcal{X}}_{\Pi_{(i)}}\|_{Frob}$. This term gives the same value independently of the mode i selected for matrix unfolding.

4 Problem Statement

In this paper, we aim to seek a decomposition sequence that minimizes the reconstruction error:

Problem 1 (Tensor Train Decomposition Sequence Selection) *Let us be given a d-dimensional tensor, $\mathcal{X} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_d}$, and a target decomposition rank, r. Our goal is to find a permutation, $\Pi = \langle \pi_1, \pi_2, \ldots, \pi_d \rangle$, which minimizes the approximation error; i.e., $argmin_{\Pi \in \mathfrak{P}} \left(Error(\hat{\mathcal{X}}_{\Pi}, \mathcal{X}) \right)$, where $\mathfrak{P}$ denotes the set of all possible $d!$ permutations.*

5 GTT: Guiding the Tensor Train

In this paper, we propose a novel approach to *guide the tensor trains* (GTT) in selecting the decomposition sequence. GTT leverages the various characteristics/statistics of the input data tensor (sparse or dense) to identify and recommend a mode ordering for the TT-decomposition process.

5.1 Data Characteristics

In this subsection, we describe data characteristics, or *features*, relevant for tensor train mode sequence selection. Note that these data characteristics are very general and can be computed for any data set with categorical entries. We leave the extension to non-categorical data to future work.
Mode Length. Given a d-mode tensor, $\mathcal{X} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_d}$, we compute the average of the mode lengths, along with the absolute and relative standard deviations:

$$\mu_{length}(\mathcal{X}) = average(l_1, l_2, \ldots, l_d), \tag{1}$$
$$\sigma_{length}(\mathcal{X}) = stdev(l_1, l_2, \ldots, l_d), \tag{2}$$
$$\phi_{length}(\mathcal{X}) = \sigma_{length} / \mu_{length}. \tag{3}$$

Intuitively, the larger the lengths of the modes, the larger will be the number of parameters to be sought. The absolute and relative standard deviations indicate how discriminative the mode length feature is in the given tensor.
Mode Entropy. Given a data set with d modes, let X_i be a discrete random variable with possible values $\{x_1, \ldots, x_{n_i}\}$ for mode i. Given this, we can compute the Entropy for mode i as $H_i = H(X_i) = -\sum_{j=1}^{n_i} p_i(j) \log_2 p_i(j)$, where $p_i(j)$ represents the probability that x_j occurs in the given mode i. Given the entropy statistics for each mode of the tensor, we then compute the average and standard deviation statistics as follows:

$$\mu_{entropy}(\mathcal{X}) = average(H_1, H_2, \ldots, H_d), \tag{4}$$
$$\sigma_{entropy}(\mathcal{X}) = stdev(H_1, H_2, \ldots, H_d), \tag{5}$$
$$\phi_{entropy}(\mathcal{X}) = \sigma_{entropy} / \mu_{entropy}. \tag{6}$$

Intuitively, entropy indicates how easy it is to have a low-rank approximation of a tensor along a given mode and the absolute and relative standard deviations indicate how discriminative the mode entropy feature is.

Tensor Density. Note that the above definition of entropy is meaningful especially for sparse tensors[3]. Therefore, we also compute a *density* statistic. Given a d-mode tensor $\mathcal{X} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_d}$, we compute the density ρ of $\mathcal{X}$ as

$$\rho(\mathcal{X}) = \frac{\#\ of\ nonzero\ values\ in\ \mathcal{X}}{l_1 \times l_2 \times \cdots \times l_d}. \tag{7}$$

Pairwise Average Conditional Entropy. The tensor train representation links consecutive modes in the sequence; therefore, pairwise statistics may also be needed. Given a data set with a d-mode tensor, let X_i be a discrete random variable with possible values $\{x_1, \ldots, x_{n_i}\}$ for mode i. The conditional entropy of X_i given X_j is defined as:

$$H_{i|j} = H(X_i|X_j) = \sum_{h=1}^{n_j} p_j(x_h) H(X_i|X_j = x_h). \tag{8}$$

We compute average pairwise conditional entropy as $ACE_{(i,j)} = \frac{H_{i|j}+H_{j|i}}{2}$. Given this, we can then compute the average and standard statistics for ACE as follows:

$$\mu_{ace}(\mathcal{X}) = average(ACE_{(i,j)} \mid i \neq j), \tag{9}$$

$$\sigma_{ace}(\mathcal{X}) = stdev(ACE_{(i,j)} \mid i \neq j), \tag{10}$$

$$\phi_{ace}(\mathcal{X}) = \sigma_{ace}/\mu_{ace}. \tag{11}$$

Note that at each step of the TT-decomposition process, the algorithm creates a core that links two modes of the tensor. Intuitively, the average pairwise entropy (ACE) indicates the ease with which one can obtain the low-rank decomposition of a pair of modes. The average and standard deviation statistics then indicate how significant this feature is in the data and how discriminative the feature is to help select pairs of modes to consider in sequence.

Pairwise Mutual Information. A related measure to conditional entropy is the pairwise mutual information. Given a d-mode tensor, let X_i be a discrete random variable with possible values $\{x_1, \ldots, x_{n_i}\}$ for mode i. The mutual information of X_i and X_j is defined as

$$MI_{(i,j)} = \sum_{x \in X_i} \sum_{y \in X_j} p_{(X_i,X_j)}(x,y) \log\Big(\frac{p_{(X_i,X_j)}(x,y)}{p_{X_i}(x) p_{X_j}(y)}\Big) \tag{12}$$

$$= H_i - H_{i|j} = H_j - H_{j|i}. \tag{13}$$

where $p_{(X_i,X_j)}$ is the joint probability mass function of X_i and X_j. We then compute that average and standard statistics for mutual information as follows:

$$\mu_{mi}(\mathcal{X}) = average(MI_{(i,j)} \mid i \neq j), \tag{14}$$

$$\sigma_{mi}(\mathcal{X}) = stdev(MI_{(i,j)} \mid i \neq j), \tag{15}$$

$$\phi_{mi}(\mathcal{X}) = \sigma_{mi}/\mu_{mi}. \tag{16}$$

Intuitively, mutual information can be used to measure how closely related the rows and columns of a given matrix are; the more closely related two modes are, the better are the chances to obtain a more accurate decomposition.

[3] Alternative definitions of entropy may be used for dense tensors.

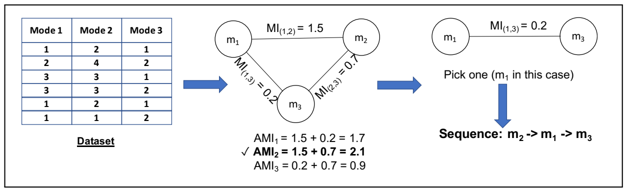


Fig. 3. GTT-AMI computation for a 3-mode tensor

5.2 GTT-NP: Number of Parameters

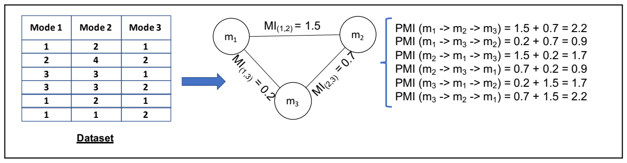
Consider the TT-decomposition process depicted in Figure 2. Here a 3-mode input tensor $\mathcal{X} \in \mathbb{R}^{l_1 \times l_2 \times l_3}$ is being converted into TT-format with a given decomposition sequence ($mode_1 \rightarrow mode_2 \rightarrow mode_3$) following Algorithm 1. In this example, the total number of parameters that the two SVD algorithms involved in the process have to solve for is the sum of the number of variables for U, SV^T, U' and SV'^T, which is $(r_0 \times l_1 \times r_1) + (r_1 \times l_2 \times l_3) + (r_1 \times l_2 \times r_2) + (r_2 \times l_3)$. It is easy to generalize this to

$$NP_\Pi(\mathcal{X}) = \sum_{i=1}^{d-1} \left(\underbrace{r_{i-1} \times l_{\pi_i} \times r_i}_{U} + \underbrace{r_i \times \prod_{j=i+1}^{d} l_{\pi_j}}_{SV^T} \right).$$

GTT-NP computes the number, $NP_\Pi(\mathcal{X})$ of parameters for each possible permutation, Π, and selects an order with the least number of parameters.

5.3 GTT-AMI and GTT-PMI: Mutual Information

Aggregate Mutual Information (AMI). Mutual information (Equation 12) can be seen as a measure of dependency between the two variables. *GTT-AMI* guides the TT-decomposition process based on the *aggregate mutual information* each mode has with the rest of the modes in the tensor. More specifically, given a d-mode tensor, the AMI value for mode i is computed as $AMI_i = \sum_{j=1}^{d} MI_{(i,j)}$. A potential strategy to guide the ordering of the modes in the TT-decomposition would be to (a) first find the mode with the largest AMI value and (b) then select this as the first mode. The process is, then, continued by (c) recomputing the AMI values among the remaining modes, (d) finding the mode with the largest (updated) AMI value among the remaining modes, and (e) selecting this as the next mode in the sequence. The process is repeated until all the modes have been ordered (when only two modes remain, the order is picked randomly). Figure 3 illustrates an example for a 3-mode (Mode 1: m_1, Mode 2: m_2, Mode 3: m_3) categorical data set. First, we compute AMI for each mode, which are: $AMI_1 = 1.5 + 0.2 = 1.7$, $AMI_2 = 1.5 + 0.7 = 2.1$, and $AMI_3 = 0.2 + 0.7 = 0.9$. In this case, AMI strategy described above would select mode m_2 as the first mode followed by m_1 or m_3. Intuitively, this process ensures that, at each step of the process, we consider and factorize a matrix where the rows have the highest statistical dependency with the columns.

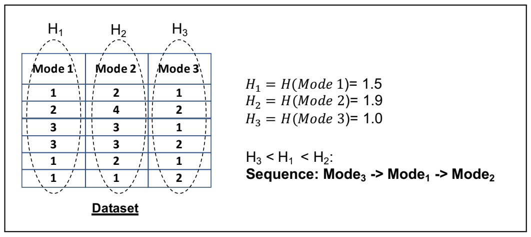


Mode 1	Mode 2	Mode 3
1	2	1
2	4	2
3	3	1
3	3	2
1	2	1
1	1	2

Dataset

PMI (m_1 -> m_2 -> m_3) = 1.5 + 0.7 = 2.2
PMI (m_1 -> m_3 -> m_2) = 0.2 + 0.7 = 0.9
PMI (m_2 -> m_1 -> m_3) = 1.5 + 0.2 = 1.7
PMI (m_2 -> m_3 -> m_1) = 0.7 + 0.2 = 0.9
PMI (m_3 -> m_1 -> m_2) = 0.2 + 1.5 = 1.7
PMI (m_3 -> m_2 -> m_1) = 0.7 + 1.5 = 2.2

Fig. 4. GTT-PMI computation for a 3-mode tensor



Mode 1	Mode 2	Mode 3
1	2	1
2	4	2
3	3	1
3	3	2
1	2	1
1	1	2

Dataset

$H_1 = H(Mode\ 1)$= 1.5
$H_2 = H(Mode\ 2)$= 1.9
$H_3 = H(Mode\ 3)$= 1.0

$H_3 < H_1 < H_2$:
Sequence: $Mode_3$ -> $Mode_1$ -> $Mode_2$

Fig. 5. GTT-IE computation for a 3-mode tensor

Path Mutual Information (PMI). Note that the above process, which first picks the mode with the highest aggregate mutual information with the rest of the modes, is likely to lead to orderings where the total mutual information along the sequence is low: Figure 4 illustrates an example, where $MI_{(1,2)} = 1.5$, $MI_{(1,3)} = 0.2$, and $MI_{(2,3)} = 0.7$. With a total MI of $(1.5 + 0.7) = 2.2$, the orders $m_1 \rightarrow m_2 \rightarrow m_3$ and $m_3 \rightarrow m_2 \rightarrow m_1$ have the highest total mutual information. In fact, *surprisingly*, permutations with a low total MI tend to lead to higher accuracies than orders with a high total MI. This somewhat counter-intuitive result (which we experimentally validate in the "Experimental Results" section), indicates that the accuracies of initial decomposition steps are very important in obtaining high accuracy in TT-decompositions. We refer to this strategy as *path mutual information (GTT-PMI).*

5.4 GTT-IE: (Inverse) Entropy

Remember that at the first step of the process, we matricize the given tensor $\mathcal{X}$ and then apply SVD to obtain U and SV^T matrices: here U represents clusters along the first selected mode and SV^T represents tensor $\mathcal{X}$ except the first mode. In the following steps of the algorithm, we apply several other clustering steps on the remaining matrix SV^T. It is therefore important that the matrix SV^T lends itself to a good clustering. One indicator of this is the entropy: if SV^T has high entropy, it is likely that it will lead to better clusters. Since the overall entropy in $\mathcal{X}$ is fixed, this implies that the matrix U should ideally have low entropy.

This leads to a third strategy, *GTT-IE*, which guides the TT-decomposition process based on the *(inverse) entropy of* each mode: at each step the algorithm selects the mode with the lowest entropy among the remaining modes. Again, Figure 5 illustrates an example of *GTT-IE*, given a 3-mode (m_1, m_2, m_3) categorical data set, IE strategy computes the entropy for each mode (H_1, H_2, H_3), and then decides a TT-decomposition sequence base on entropy in ascending order.

5.5 GTT-HYB: Hybrid Strategy

In Table 3, we list the data sets we use in our experiments along with along with the (non-hybrid) strategy with the best accuracy performance. As we see in the table, none of the strategies lead to a universally accurate order. While this is initially disappointing, the facts that different strategies work well for different data sets and that, often, where one strategy fails to lead to an accurate decomposition, another strategy excels, indicate that a hybrid strategy which carefully switches between the different approaches can lead to a better accuracy than any of the individual strategies.

To show the feasibility of such a hybrid technique, for each strategy[4], $\mathfrak{S}$, we have considered the data characteristics described earlier Section 5.1 as features and train a (linear) SVM classifier (with L1-regularization) that separates the data sets for which the strategy provides better accuracies than the rest (i.e., strategy $\mathfrak{S}$ vs. rest). In particular, for each scenario we consider the top-20% of the tensor instances for which the given strategy returns the best results against the lowest-20% of the tensor instances for which the given strategy returns the worst results. Intuitively, the separator can be interpreted as a feature selector that describes the data characteristics that best matches the given strategy. For each decomposition scenario, we then select the strategy that is recommended collectively by the trained separators; for any scenario for which the classifiers recommend more than one strategy, we pick the strategy that has the largest margin from the corresponding separator.

5.6 Complexity of GTT Decomposition

Let $\mathcal{X}$ be a d-mode input tensor and $t = |\mathcal{X}|$ indicates the number of non-zero entries in data set. Let also n_i denote the size of mode d_i and n denote the average mode size.

Guidance Step. The time complexities for the various strategies are as follows:

- *GTT-AMI* makes a pass over t data and for each it computes its contribution to the mutual information among $\frac{d(d-1)}{2}$ mode pairs; therefore its cost is $O\left(t \times \frac{d(d-1)}{2}\right)$.
- *GTT-PMI* also computes mutual information for all pairs of modes, but then it further computes a minimum path on the resulting graph with d nodes and $\frac{d(d-1)}{2}$ edges; therefore its cost is $O\left(\left(t \times \frac{d(d-1)}{2}\right) + \left(\frac{d(d-1)}{2} + d \log d\right)\right)$.
- *GTT-NP* enumerates $d!$ many sequences and, for each sequence computes the corresponding number of variables at $O(d)$ time – therefore it costs $O(d! \times d)$.
- *GTT-IE* requires one pass over the entire data for computing all of the mode entropies – i.e., its cost is $O(t)$.

[4] Note that the two mutual information based strategies, *GTT-AMI* and *GTT-PMI*, are hard to separate; since, as we see in Tables 4 and 5 in Section 6, *GTT-PMI* is overall more accurate among the two, we omit emphGTT-AMI in hybrid selection.

Table 2. Data sets [9]

Data set	#Inst.	#Modes	Data set	#Inst.	#Modes
dermatology	366	34	flare	1395	11
mushroom	8124	23	house-votes	435	17
soybean	307	36	tic-tac-toe	958	10
breast	699	10	nursery	12960	9
balance-scale	625	5	primary-tumor	339	18
hayes-roth	160	6	lymphography	148	19
car	172	7	spect	267	23
chess	3196	37			

Note that, as we experimentally show in the next section (Table 4), the time complexity for statistics collection is negligible relative to the time needed to decompose the tensor.
Decomposition Step. GTT provides a decomposition order which is then fed into TT-SVD to obtain the actual decomposition. The decomposition time complexity is therefore equal to that of TT-SVD[23], which is $O(dnr^3)$ and the number of parameters will be $O(dnr + (d-2)r^3)$.

6 Experimental Results

Here, we present experimental evaluations of the proposed GTT strategies[5]. Note that (once the decomposition order is selected) the data tensors are decomposed using TT-SVD [23] on a 4-core CPU (2.7GHz each) machine, with 16GB RAM.

6.1 Competitors

We compare five order selection strategies (*GTT-AMI*, *GTT-PMI*, *GTT-IE*, *GTT-NP*, *GTT-HYB*) and a baseline strategy, ARB, which represents the "average" decomposition performance of uninformed (i.e. arbitrary) order selection.
Evaluation Criteria. For accuracy, we adapt the reconstruction error introduced in Section 3. We report and compare average reconstruction errors for each strategy and the percentage improvement over ARB:

– Given a d-mode tensor, we enumerate ALL ($d!$) permutations and compute error for each permutation.
– We use the mean of all these $d!$ reconstruction errors as the (average) error for arbitrary selection, ARB.

In addition to the absolute values of reconstruction errors, we also report percentages of decompositions with better than (B) and worse than (W) the average ranking by arbitrary selection, ARB. We further report the ratio $gain = B/W$ – the value of $gain$ indicates how well a given strategy promotes good decomposition, while avoiding the bad ones.

We also report the average decomposition times for the decomposition orders selected by the various strategies.

[5] Our implementation and data sets can be found: https://shorturl.at/DMOSY

Table 3. The relative average ranking against ARB for each data set (we normalize average ranking of ARB strategy as 1, and the bold number means the best ranking within four proposed strategies - **the lower, the better**) and the percentage improvement in reconstruction error (RE % impr.) against ARB using the *GTT-HYB* strategy - **the higher, the better**. *Inst. weighted average = (# of instances for a data set * Relative average ranking or RE % impr. for a strategy)/(total # of instances).

	Relative average ranking (Lower, the better)								**RE % impr. using HYB (Higher, the better)**	
	r=3				r=5				r=3	r=5
Data set	IE	NP	PMI	AMI	IE	NP	PMI	AMI	% impr.	% impr.
tic-tac-toe	**0.68**	1.04	0.97	0.93	**0.67**	1.04	0.98	0.94	45%	49%
balance-scale	**0.56**	1.12	0.97	0.71	**0.55**	1.02	0.93	0.71	16%	16%
breast	**0.75**	1.01	0.99	0.92	**0.74**	1.03	0.97	0.94	10%	12%
hayes-roth	**0.66**	0.75	0.75	0.75	**0.65**	0.73	0.74	0.69	4%	10%
primary-tumor	0.97	**0.68**	0.92	0.93	0.98	**0.66**	0.90	0.93	10%	5%
nursery	0.96	**0.79**	0.92	0.82	1.00	0.90	0.93	**0.81**	4%	5%
dermatology	**0.82**	0.96	1.01	1.12	**0.84**	0.96	0.99	1.13	4%	4%
spect	0.98	1.00	0.96	**0.95**	0.97	1.00	0.95	**0.94**	8%	4%
flare	**0.83**	0.86	0.96	1.10	**0.84**	0.85	0.96	1.11	4%	3%
house_votes	**0.92**	1.05	0.99	1.21	**0.89**	1.06	1.00	1.22	2%	3%
soybean	0.92	**0.88**	0.97	1.02	0.96	**0.92**	0.97	0.99	2%	3%
mushroom	1.00	0.90	**0.87**	1.12	1.05	**0.85**	0.90	1.13	3%	3%
lymphography	**0.77**	0.88	0.97	1.03	**0.77**	0.92	0.97	1.03	5%	1%
car	1.10	0.87	0.98	**0.66**	1.15	1.14	0.99	**0.71**	0%	-2%
chess	**0.91**	0.97	0.98	1.02	**0.90**	0.96	0.98	1.02	-16%	-6%
Average	**0.86**	0.92	0.95	0.95	**0.86**	0.94	0.94	0.95	6.7%	7%
*Inst. weighted Average	**0.88**	0.91	0.96	1.00	**0.89**	0.93	0.96	1.01	7%	7%

Data Sets. Table 2 lists the 15 data sets we use in these experiments. The data sets are taken from the UCI Machine learning repository [9]. From each data set, we extracted randomly selected 3-, 4-, and 5-mode tensor instances (up to 100 each, as allowed by the dimensionality of the data set). The total number of tensors extracted from these data sets and used in the experiments is 3632.

Target Ranks. Here, we consider two target ranks, 3 and 5. As discussed in Section 3, we assume the target TT-rank is given and fixed for each mode. While there are several non-parametric decomposition techniques, such as [13] which can learn also the appropriate rank, this is outside of the scope of this paper. We leave this to the future works.

6.2 Evaluations and Analysis

Accuracy. In Table 3, we first list the relative average ranking for each proposed strategy against ARB (lower, the better), as we can see, the best single strategy can vary from data set to data set – this motivates the need for a hybrid strategy (*GTT-HYB*) to select an effective combined strategy. As shown in Table 3, *GTT-HYB* provides improvements for all data set except the *car* and *chess* data sets. To get a more general view of the benefit of proposed strategies, in Table 4, we aggregate all data sets and report average reconstruction errors and percentage of improvements against the baseline (ARB). As we see in the table, all proposed GTT strategies improve reconstruction performance against ARB, with *GTT-IE* providing the highest improvement among the single criterion strategies. The

Table 4. Average reconstruction error, rate of improvement against arbitrary selection (ARB) and average decomposition time.

	Average Reconstruction Error (Lower, the better)		**Rate of Improvement (Higher, the better)**		**Avg. Dec. Time (ms)**	
Method	r=3	r=5	r=3	r=5	r=3	r=5
ARB	5.18	5.37	-	-	82.9	85.3
IE	4.94	5.1	4.7%	5.0%	**78.6**	**81.8**
NP	5.07	5.29	2.1%	1.5%	84.4	82.5
PMI	5.1	5.29	1.6%	1.5%	81.5	84.3
AMI	5.14	5.34	0.8%	0.4%	82.1	82.3
HYB	**4.86**	**5.05**	**6.2%**	**6.0%**	80.7	82.3

Table 5. Percentages of decompositions with better than (B) and worse than (W) the rank of decomposition returned on average by an uniformed, arbitrary ARB selection strategy)

Method	**r=3**			**r=5**		
	B	W	$gain$	B	W	$gain$
IE	54.0	34.0	1.6	53.0	35.0	1.5
NP	38.0	25.0	1.5	37.0	27.0	1.4
PMI	46.0	34.0	1.4	46.0	34.0	1.4
AMI	45.0	43.0	1.0	45.0	42.0	1.1
HYB	54.5	29.2	**1.9**	52.4	30.8	**1.7**

table also shows that the hybrid strategy (*GTT-HYB*, described in Section 5.5) provides the highest overall improvement in accuracy. Table 3 also depicts the percentage improvement of reconstruction error (RE) against ARB using the *GTT-HYB* strategy for each data set, and we further see that the proposed hybrid strategy is indeed beneficial for 13 out of 15 of the considered data sets.

Again, with aggregating all data sets, in Table 5, we report the percentage of tensors for which each strategy returns better than (B) and worse than (W) the arbitrary selection, ARB, and the overall gain ($gain = B/W$). As we see, the *GTT-IE* strategy provides the largest *gain* among the four strategies and as before *GTT-HYB* strategy provides the best overall gain for both target tt-ranks.

Note that, among the two mutual information, based strategies, *GTT-PMI* is more effective than *GTT-AMI* in terms of both reconstruction error (Table 4) and *gain* (Table 5). Therefore, as reported in Section 5.5, we do not consider GTT-AMI, when constructing a hybrid strategy.

Decomposition Time. Table 4 reports the average decomposition times for different strategies. As we discussed in Section 5.6, the proposed strategies do not add any overhead to the decomposition time over arbitrary selection, ARB. In fact, the hybrid strategy, *GTT-HYB*, appears to reduce the decomposition time over ARB. While this is not our focus in this paper, we plan to explore this further in future work.

Top-Contributors to Each Strategy. In Table 6, we present the top-3 positive and/or negative contributors (among the various statistics considered in Section 5.1) for the *GTT-IE*, *GTT-NP*, and *GTT-PMI* strategies:

Table 6. Three major contributors to the *GTT-IE*, *GTT-NP*, and *GTT-PMI* strategies (positive values indicate positive, negative values indicate negative contribution)

IE	$\sigma_{ace}[3.9]$; $\phi_{ace}[-2.7]$; $\rho[-1.9]$
NP	$\phi_{length}[3.1]$; $\rho[-2.0]$; $\sigma_{entropy}[-1.1]$
PMI	$\sigma_{ace}[3.2]$; $\phi_{ace}[-2.0]$; $\mu_{length}[1.8]$

– For *GTT-IE*, the two main contributors are σ_{ace} and ϕ_{ace}. This echos the argument in Section 5.4: *GTT-IE* prefers that the entropies of the modes are considered in ascending order and thus *GTT-IE* is more effective when the discriminatory power of ACE is high.
– As discussed in Section 5.2, the number of parameters that needs to be learned depends on the length of the modes and the more discriminative the mode length parameter is, the more effective *GTT-NP* – this explains the positive contribution of ϕ_{length} to the *GTT-NP* selection criterion.
– For the mutual information based strategy, *GTT-PMI*, the higher the spread of ACE, the higher the impact of *GTT-PMI*. This confirms our discussion in Section 5.3: mutual information can be considered as a measure of dependency and, since the entropy of a mode is fixed, its dependency with the adjacent mode (mutual information) is constrained by the conditional entropy between them. Hence, the more the parameter ACE is (i.e., the larger is the value of σ_{ace}), the higher the benefits of *GTT-PMI*.

7 Conclusion

While the TT-decomposition promises a good trade-off between accuracy and resource requirements, the final accuracy is highly dependent on the order of the tensor modes in the tensor train. In this paper, we proposed a novel approach for *guiding the tensor train* (GTT) in selecting the decomposition sequence. We have shown that we can leverage the various characteristics of the given data set to identify an effective order strategy. In particular, we proposed three order selection strategies and have shown that a *hybrid (HYB)* strategy that combines these three strategies taking into account the specific characteristics of the given data set can lead to decomposition sequences with high accuracy.

References

1. Batselier, K.: The trouble with tensor ring decompositions. CoRR abs/1811.03813 (2018)
2. Batselier, K., Yu, W., Daniel, L., Wong, N.: Computing low-rank approximations of large-scale matrices with the tensor network randomized svd. SIAM Journal on Matrix Analysis and Applications **39**(3), 1221–1244 (2018)
3. Bedo, M.V.N., Ciaccia, P., Martinenghi, D., de Oliveira, D.: A k-skyband approach for feature selection. In: Similarity Search and Applications - 12th International Conference, SISAP 2019, pp. 160–168 (2019)
4. Candan, K.S., Sapino, M.L.: Data Management for Multimedia Retrieval. Cambridge University Press, USA (2010)

5. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. Psychometrika 35(3), 283–319 (Sep 1970). https://doi.org/10.1007/BF02310791
6. Chen, Y., *et al.*: Sharing residual units through collective tensor factorization to improve deep neural networks. In: IJCAI-18. pp. 635–641. (7 2018)
7. Dash, M., Choi, K., Scheuermann, P., Huan Liu: Feature selection for clustering - a filter solution. In: 2002 IEEE ICDM, 2002. Proceedings. pp. 115–122 (Dec 2002)
8. Dash, M., Liu, H., Yao, J.: Dimensionality reduction of unsupervised data. In: Proceedings Ninth IEEE Inter. Conference on Tools with Artificial Intelligence. pp. 532–539 (1997).
9. Dua, D., Graff, C.: UCI machine learning repository (2017)
10. Harshman, R.: Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. UCLA Working Papers in Phonetics 16 (1970)
11. Houle, M.E., Kashima, H., Nett, M.: Fast similarity computation in factorized tensors. In: Similarity Search and Applications - 5th International Conference, SISAP 2012, pp. 226–239 (2012)
12. Huang, S., Candan, K.S., Sapino, M.L.: Bicp: Block-incremental cp decomposition with update sensitive refinement. In: CIKM'16. pp. 1221–1230. CIKM '16, (2016).
13. Imaizumi, M., Maehara, T., Hayashi, K.: On tensor train rank minimization : Statistical efficiency and scalable algorithm. In: NIPS'17, pp. 3930–3939. (2017)
14. Jeon, I., Papalexakis, E.E., Kang, U., Faloutsos, C.: Haten2: Billion-scale tensor decompositions. In: 2015 IEEE 31st ICDE. pp. 1047–1058 (April 2015)
15. Kim, M., Candan, K.S.: Decomposition-by-normalization (dbn): Leveraging approximate functional dependencies for efficient cp and tucker decompositions. Data Mining and Knowledge Discovery 30(1), 1–46 (Jan 2016).
16. Ko, C.Y., Lin, R., Li, S., Wong, N.: Misc: Mixed strategies crowdsourcing. In: IJCAI-19. pp. 1394–1400. IJCAI (7 2019). https://doi.org/10.24963/ijcai.2019/193
17. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(1), 273 – 324 (1997).
18. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. 51(3), 455–500 (Aug 2009).
19. Li, L., Yu, W., Batselier, K.: Faster tensor train decomposition for sparse data. ArXiv (2019)
20. Mickelin, O., Karaman, S.: Tensor ring decomposition. CoRR abs/1807.02513 (2018)
21. Novikov, A., Podoprikhin, D., Osokin, A., Vetrov, D.: Tensorizing neural networks. In: NIPS'15. Volume 1. pp. 442–450. NIPS'15, (2015)
22. Novikov, A., Trofimov, M., Oseledets, I.: Exponential Machines. arXiv e-prints arXiv:1605.03795 (May 2016)
23. Oseledets, I.: Tensor-train decomposition. SIAM Journal on Scientific Computing 33(5), 2295–2317 (2011).
24. Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika 31(3), 279–311 (1966)
25. Yamaguchi, Y., Hayashi, K.: Tensor decomposition with missing indices. In: IJCAI'17, 3217–3223. IJCAI'17, (2017)
26. Yu, L., Liu, H.: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: ICML'03 . vol. 2, pp. 856–863 (2003)
27. Zhao, Q., Zhou, G., Xie, S., Zhang, L., Cichocki, A.: Tensor ring decomposition. CoRR abs/1606.05535 (2016)