GTT: Leveraging Data Characteristics for Guiding the Tensor Train Decomposition*

Mao-Lin Li, K Selçuk Candan[1]

Arizona State University, Tempe AZ, USA

Maria Luisa Sapino[2]

University of Turino, Turino, Italy

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

*This work is an extended version of *Mao-Lin Li, K. Selçuk Candan, Maria Luisa Sapino. "GTT: Guiding the Tensor Train Decomposition" published in the International Conference on Similiarity Search and Applications (SISAP) 2020.* This work is supported by NSF#1610282 "DataStorm: A Data Enabled System for End-to-End Disaster Planning and Response", NSF#1633381 "BIGDATA: Discovering Context-Sensitive Impact in Complex Systems", NSF#1909555 "pCAR: Discovering and Leveraging Plausibly Causal (p-causal) Relationships to Understand Complex Dynamic Systems", and "FourCmodeling": EUH2020 Marie Sklodowska-Curie grant agreement No 690817. Results were obtained using the ChameleonCloud resources supported by the NSF.

[1]Arizona State University, {maolinli@asu.edu, candan@asu.edu}

[2]University of Turino, {mlsapino@di.unito.it}

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, Order selection.

1. Introduction

Tensors are commonly used to represent multi-dimensional sets. Consequently, tensor decomposition operations, such as CP [1] [2] and Tucker [3] form the basis of many AI techniques for data analysis and knowledge discovery. 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 any applications [4] [5], but due to the denseness of the core, it also is subject to exponential memory growth with the number of tensor modes. The tensor train (TT) decomposition addresses this problem, by seeking a sequence of 3-mode cores [6]: 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, including deep learning [7][8], crowdsourcing [9] and recommendation systems [10].

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 [12]. 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

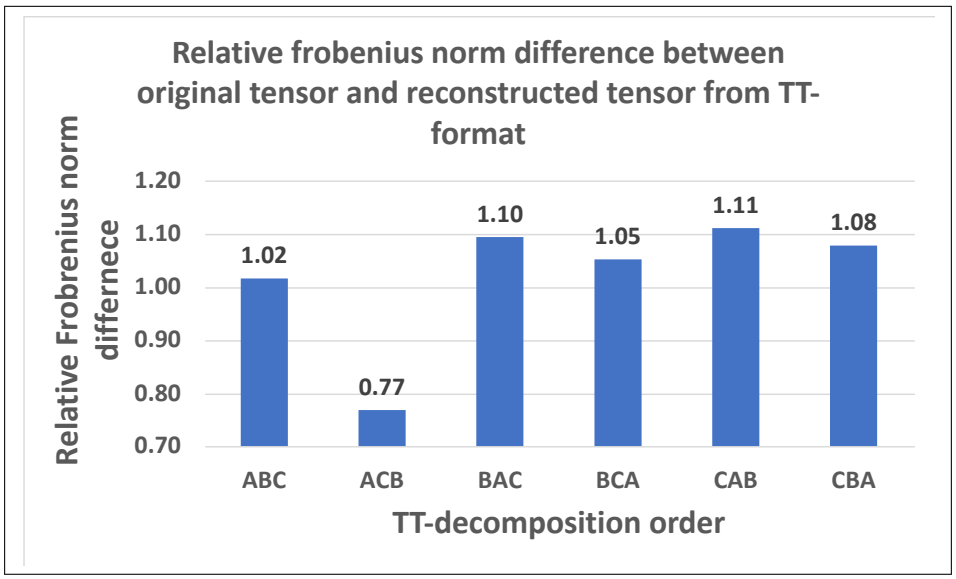

Figure 1: Effect of the decomposition order on the accuracy for a 3-mode tensor from the Wisconsin Diagnostic Breast Cancer data [11]: $ID(mode_A)$, $Diagnosis(mode_B)$ and $Radius(mode_C)$. See Section 7 for more details

Machine Learning Repository [11]; 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 our preliminary work [13], we proposed a novel approach for *guiding the tensor train* (GTT) in selecting the mode sequence for tensor train decompositions for *categorical data sets*. More specifically,

- 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 given categorical-valued data set.

In this paper, we extend the GTT technique to data sets with continouous/non-categorical attributes. We further introduce two statistics collection strategies *statistics-then-discretization (StD)* and *discretization-then-statistics (DtS)* to encode continuous-valued data in the form of a tensor for analysis. Experiments reported in Section 7 show that the proposed HYB strategy provides an effective order selection strategy for both categorical and continuous valued, without any additional decomposition time overhead.

1.3. Organization of the Paper

This paper is organized as follows: In the next section, we present the related work for tensor decomposition techniques. Section 3 presents the relevant notations and the background for the tensor train decomposition. Section 4 describes the problem statement we tackle. Section 5 describes the data characteristics we extract for the proposed method: *guide the tensor trains* (GTT) in Section 6. Section 7 experimentally evaluates the effectiveness of GTT. And then we conclude the paper in Section 8.

2. Related Work

2.1. Tensor

The tensor model maps a multi-attribute schema into an d-modal array. More formally, let l_j denote the number of distinct values that the j^{th} attribute (or the j^{th} mode) can take. The tensor $\mathcal{X}$ is then an d-modal array such that $\mathcal{X} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_d}$. Intuitively, the modes of the tensor represent different $factors$

that impact an observation and the value that the tensor records for a given cell corresponds to an observation for a specific combination of factor instances. *Tensor unfolding*, or *matrization*, is one of fundamental operation for tensor methods. Considering a tensor as a multi-modal array, unfolding it consists of reading its element in such a way as to obtain a matrix instead of tensor. Mode-i unfolding is obtained by considering the i^{th} mode as the first dimension of a matrix and collapsing the other into the other dimension of that matrix. For a tensor of size $(l_1 \times l_2 \times \cdots \times l_d)$, the mode-$i$ unfolding of this tensor will be the size $(l_i, l_1 \times \cdots \times l_{i-1} \times l_{i+1} \times \cdots \times l_d)$.

2.2. Tensor Decomposition

Tensor decomposition has been shown to be effective in multi-aspect data analysis for capturing high-order structure in high-dimensional data [4]. The CANDECOMP/PARAFAC (CP) and the Tucker [14] are the two most popular tensor decomposition algorithms. The CP decomposition factorizes the tensor into r component matrices (where r is a user supplied non-zero integer value also referred to as the *rank* of the decomposition). The Tucker decomposition generalizes singular value matrix decomposition (SVD) to high dimensional data. However, a major challenge is its high computational complexity and large memory overhead. There are several parallel and block-based implementations to alleviate this issue, such as GridParafac [15], GigaTensor [16], HaTen2 [17], BICP [18].

2.3. Tensor Train Decomposition

Tensor-train decomposition [6] provides a memory-saving representation called *TT-format*, , which preserves the representation power. For example, Given a d-modal tensor, the space complexity of traditional tensor decomposition (e.g. Tucker) is exponential in d, whereas TT-format has a with linear space complexity by creating a linear tensor network (see Figure 2).

TNrSVD [19] adapts the randomized SVD to implement TT-decomposition, and FastTT [20] computes the TT-decomposition of a sparse tensor by its sparsity. However, as discussed in the introduction, TT-decomposition involves

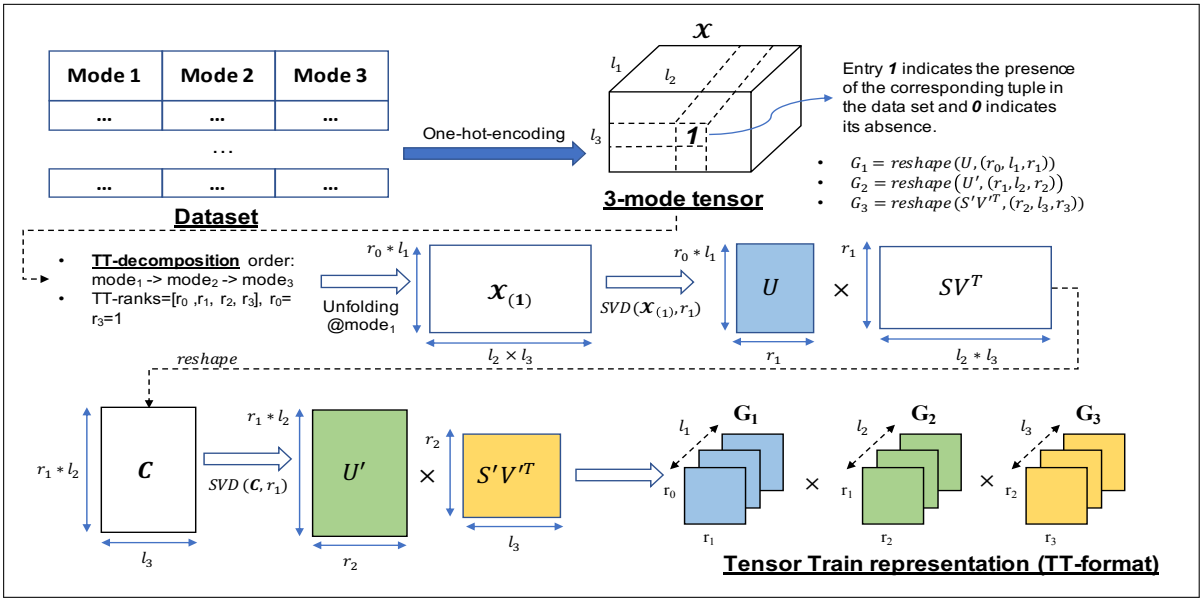

Figure 2: An example of TT-decompostion for converting a 3-mode tensor $\mathcal{X}^{I_1 \times I_2 \times I_3}$ into *TT-format*.

strictly sequential multi-linear products over latent cores and this makes it difficult to search for best TT representation for a given tensor. [21] and [12] 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 [22], however, it does not eliminate the need to pick a (*circularly-arranged*) permutation of modes.

2.4. Feature Selection in High Dimensional Data

Feature selection techniques, such as [23] [24], search for the most relevant attributes of the data set (for a given application) to reduce the dimensionality, for example, *Entropy* tends to be low for data that contain tight clusters [25, 26]. 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.

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
π_z	The mode in the z^{th} position in a sequence of a given tensor train decomposition
Π	A permutation of modes $\langle \pi_1, \pi_2, \ldots, \pi_d \rangle$
r_i	TT-rank of mode i
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 of the random variable for mode i given the random variable for 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

2.5. Guiding the Tensor Train Decomposition (GTT)

Inspired by above researches, in our preliminary work [13], we proposed GTT, which leverages various data properties (e.g. mode entropy, pair-wise mutual information) of high-dimensional and categorical data sets as a guideline to select an effective sequence for tensor train decomposition. In this paper, we extend these results to continuous valued data and consider alternative discretization strategies for tensor-encoding of continuous valued data sets.

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 [6] is obtained by applying a sequence of singular value decompositions (SVD) to approximate the original tensor: given

Algorithm 1 TT-SVD (adapted from [6])

Input:

A permutation $\Pi = \langle \pi_1, \pi_2, \ldots, \pi_d \rangle$, of modes;

A d-mode tensor $\mathcal{X} \in \mathbb{R}^{l_{\pi_1} \times l_{\pi_2} \times \cdots \times l_{\pi_d}}$;

A list of target tt-ranks, $\langle r_{\pi_0}, r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_d} \rangle$, $r_{\pi_0} = r_{\pi_d} = 1$;

Output:

TT-format with TT-cores $G_{\pi_1}, G_{\pi_2}, \ldots G_{\pi_d}$.

- $numel(C)$: number of elements in C.
- $reshape(A, [d_1, \ldots, d_k])$: reshape an array A into shape $d_1 \times d_2 \times \cdots \times d_k$.
- $min(a, b)$: return a if $a < b$, else return b.

procedure TT-SVD($\mathcal{X}, \langle r_{\pi_0}, r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_d} \rangle$)
    $C = \mathcal{X}$.
    for $k \leftarrow 1$ *to* $d - 1$ **do**
        $C \leftarrow\ reshape(C, [r_{\pi_{k-1}} \times l_{\pi_k}, \frac{numel(C)}{r_{\pi_{k-1}} \times l_{\pi_k}}])$.
        $U, S, V = SVD(C, r_{\pi_k} = min(r_{\pi_k}, l_{\pi_k}))$.
        $G_k \leftarrow\ reshape(U, [r_{\pi_{k-1}}, l_{\pi_k}, r_{\pi_k}])$.
        $C \leftarrow SV^T$.
    end for
    $G_{\pi_d} \leftarrow C$.
    return TT-format with TT-cores $G_{\pi_1}, \ldots, G_{\pi_d}$.
end procedure

- (i) a permutation, $\Pi = \langle \pi_1, \pi_2, \ldots, \pi_d \rangle$, of modes, where π_z is mode in the z^{th} position in a sequence of a given tensor train decomposition,
- (ii) an input tensor, $\mathcal{X} \in \mathbb{R}^{l_{\pi_1} \times l_{\pi_2} \times \cdots \times l_{\pi_d}}$,
- (iii) a sequence of decomposition ranks, $\langle r_{\pi_0}, r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_d} \rangle$, where $r_{\pi_0} = r_{\pi_d} = 1$,

the tensor train decomposition approximates the input tensor, $\mathcal{X}$, with a sequence of tensor cores $G_{\pi_k} \in \mathbb{R}^{r_{\pi_{k-1}} \times l_{\pi_k} \times r_{\pi_k}}, k = 1 \ldots d$, where $\mathcal{X} \approx \hat{\mathcal{X}}_\Pi = G_{\pi_1} \cdot G_{\pi_2} \cdot \cdots \cdot G_{\pi_d}$.

In the index form the decomposition is written as:

$$\mathcal{X} \approx \hat{\mathcal{X}}_{\Pi}(i_{\pi_1}, i_{\pi_2}, \ldots, i_{\pi_d}) = \sum_{\alpha_{\pi_0}, \alpha_{\pi_1}, \ldots, \alpha_{\pi_d}} G_{\pi_1}(\alpha_{\pi_0}, i_{\pi_1}, \alpha_{\pi_1}) \times G_{\pi_2}(\alpha_{\pi_1}, i_{\pi_2}, \alpha_{\pi_2}) \cdots \times G_{\pi_d}(\alpha_{\pi_{d-1}}, i_{\pi_d}, \alpha_{\pi_d}), \tag{1}$$

where i_{π_m} represents the index of mode π_m and $1 \leq \alpha_{\pi_m} \leq r_{\pi_m}$.

In this paper, we will assume that all ranks (except $r_0 = r_{\pi_d} = 1$) have the same value, r. Note that, while there are several non-parametric decomposition techniques, such as [27] 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. Algorithm 1 presents the pseudocode and Figure 2 visualizes the TT-SVD process for a 3-mode tensor $\mathcal{X} \in \mathbb{R}^{l_{\pi_1} \times l_{\pi_2} \times l_{\pi_3}}$, where l_i represents the size of mode i.

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, $\hat{\mathcal{X}}_{\Pi}$: $Error(\hat{\mathcal{X}}_{\Pi}, \mathcal{X}) = \frac{\|\mathcal{X}_{(i)} - \hat{\mathcal{X}}_{\Pi_{(i)}}\|_{Frob}}{\|\mathcal{X}_{(i)}\|_{Frob}}$. Note that 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 with a permutation of modes* $\langle \pi_1, \pi_2, \ldots, \pi_d \rangle$, $\mathcal{X} \in \mathbb{R}^{l_{\pi_1} \times l_{\pi_2} \times \cdots \times l_{\pi_d}}$, *and a sequence of TT-ranks,* $\langle r_{\pi_0}, r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_d} \rangle$, *where* $r_{\pi_0} = r_{\pi_d} = 1$. *Our goal is to find a permutation,* $\Pi = \langle \pi_1, \pi_2, \ldots, \pi_d \rangle$, *which minimizes the approximation error; i.e.,*

$$\Pi = \underset{\Pi \in \mathfrak{P}}{argmin} \left(Error(\hat{\mathcal{X}}_{\Pi}, \mathcal{X}) \right),$$

where $\mathfrak{P}$ *denotes the set of all possible* $d!$ *permutations.*

5. Tensor Data Characteristics to Guide Tensor Train Decomposition

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. Tensor Encoding

A high-dimensional data set can be viewed as a set of *tuple*s or a *tensor*. When the attributes of the data set are categorical, the tensor representation is easy to obtain: As we illustrated in Figure 2, a given data set with categorical entries in tuple representation can be converted to an occurrence tensor with one-hot-encoding paradigm, in which each entry with value 1 indicates the presence of the corresponding tuple in the data set and 0 indicates its absence. Note that duplicated tuples will be discarded in the tensor encoding. Nevertheless, we will keep these duplicated tuples when we compute other data characteristics in the following sections.

For data sets with *continuous* entries, however, we need to discretize the modes with continuous values into categorical values before such an encoding is possible. There are various methods for obtaining discrete representation of continuous valued data sets; these include equal-width binning, equal-frequency binning, k-means clustering, or decision trees [28]. Here, we adopt *equal-width* binning: let C_i be a set of continuous values of mode i; given a number, N_i, of bins, we compute the length, W_i, of the discretization window as

$$W_i = \frac{(max(C_i) - min(C_i))}{N_i}. \tag{2}$$

Given this window size, each entry, v, in C_i will be represented with the corresponding bin.

5.2. Statistics Collection Strategies for Continuous Valued Data

As described above, discretization is a necessary tensor encoding step for data sets with continuous entries. However, the data statistics that will be used

for guiding the tensor train decomposition process can be collected *before* or *after* the discretization process.

5.2.1. Discretization-then-Statistics (DtS)

Given a data set with continuous entries, *Discretization-then-Statistics (DtS)* strategy first categorizes continuous entries into discrete values with the method described above to generate a corresponding tensor, and then extracts data characteristics treating the data as categorical.

5.2.2. Statistics-then-Discretization (StD)

Given a data set with continuous entries, *Statistics-then-Discretization (StD)* strategy first extracts data characteristics from the continuous data and then categorizes continuous entries into discrete values to generate the corresponding tensor encoding.

We experimentally evaluate the performance of these two statistics collection strategies in Section 7.

5.3. Data Characteristics

Here, we describe data characteristics, or *features*, relevant for tensor train mode sequence selection. Let us consider a tensor with n tuples and m modes (or dimensions).

5.3.1. Mode Lengths

Given a data set with d modes, we hypothesize that the value of d will have an indirect impact on the selected order. In particular, as the value of d increases, the number of parameters that need to be solved during the decomposition process increases and different orderings may lead to different number of parameters – this may have an impact on the strategy to be used for permutation selection. Given a d-mode tensor with a permutation of modes $\langle \pi_1, \pi_2, \ldots, \pi_d \rangle$, $\mathcal{X} \in \mathbb{R}^{l_{\pi_1} \times l_{\pi_2} \times \cdots \times l_{\pi_d}}$, we compute the average of mode lengths, along with the

absolute and relative standard deviations:

$$\mu_{length}(\mathcal{X}) = average(l_{\pi_1}, l_{\pi_2}, \ldots, l_{\pi_d}), \quad (3)$$

$$\sigma_{length}(\mathcal{X}) = stdev(l_{\pi_1}, l_{\pi_2}, \ldots, l_{\pi_d}), \quad (4)$$

$$\phi_{length}(\mathcal{X}) = \sigma_{length}/\mu_{length}. \quad (5)$$

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.

5.3.2. Mode Entropy

We next argue that the entropy of the data captured by the various modes of the data may also impact the tensor train decomposition order. Intuitively, entropy would indicate 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.

Modes with Categorical Data. Given a data set with d modes, for the data with *categorical* entries, 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

$$H_i = H(X_i) = -\sum_{j=1}^{n_i} p_i(j) \log_2 p_i(j), \quad (6)$$

where $p_i(j)$ represents the probability that x_j occurs in the given mode i.

Modes with Continuous Data. For the data sets with *continuous* entries, we cannot directly apply the basic entropy definition. To quantify entropy for continuous variables, [29] extends the idea of Shannon entropy, a measurement for the level of surprise of a random variable, to continuous probability distributions through differential entropy. The actual continuous version of discrete entropy is the limiting density of discrete points (LDDP) [30] and the conventional differential entropy extended from discrete Shannon entropy [29] is a limiting case

of the LDDP and loses its fundamental association with discrete entropy. Therefore, here, we adapt the method proposed in [31], where the entropy estimator is based on the first nearest neighbor distances of the sample points: let X_i be a continuous random variable for mode i in certain metric space, i.e. there is a distance function $||x - x^{'}||$ between any two instances of X_i; the entropy of X can be estimated as:

$$H_i^* = \hat{H}^*(X_i) = -\psi(k) + \psi(N) + \log(c_d) + \frac{d}{N}\sum_{i=1}^{N}\log\epsilon(i), \tag{7}$$

where

- ψ: the digamma function.
- k: the k-nearest neighbor, we set $k = 1$ as default.
- N: the numbers of instances
- c_d: the volume of the d-dimensional unit ball.
- $\epsilon(i)$: twice the distance from x_i to its kth nearest neighbor.

The detailed proof can be found in [31, 32].

Entropy Statistics for the Tensor. Given the entropy for each mode of the tensor, we can compute the average and standard deviation statistics as follows (note that we use entropy for categorical variable H in the following formulas and sections, but entropy for continuous variable H^* can be substituted for data sets with continuous values):

$$\mu_{entropy}(\mathcal{X}) = average(H_1, H_2, \ldots, H_d), \tag{8}$$

$$\sigma_{entropy}(\mathcal{X}) = stdev(H_1, H_2, \ldots, H_d), \tag{9}$$

$$\phi_{entropy}(\mathcal{X}) = \sigma_{entropy}/\mu_{entropy}. \tag{10}$$

The absolute and relative standard deviations indicate how discriminative the mode entropy feature is in the given tensor.

5.3.3. 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

$$\rho(\mathcal{X}) = \frac{\#\ of\ nonzero\ values\ in\ \mathcal{X}}{l_1 \times l_2 \times \cdots \times l_d}. \tag{11}$$

5.3.4. Pairwise Average Conditional Entropy

The above statistics of mode length and entropy consider each mode in isolation. Yet, as we mentioned in Section 3, the tensor train representation links consecutive modes in the sequence, and we believe these links provide us extra information for the sequence. To measure the strengths of the linkages, we can abstract the given data set as a mode-graph representation and compute pairwise statistics to guide tensor train decomposition. First of these pairwise statistics is the *pairwise average conditional entropy* described below.

Mode Pairs with Categorical Data. Let X_i denote a discrete random variable with possible values $\{x_{i,1}, \ldots, x_{i,n_i}\}$ corresponding to mode i. The conditional entropy of X_i given X_j is defined as:

$$H_{i|j} = H(X_i|X_j) = \sum_{h=1}^{n_j} p_j(x_{j,h}) H(X_i|X_j = x_{j,h}). \tag{12}$$

Given this, we can compute average pairwise conditional entropy as $ACE_{(i,j)} = \frac{H_{i|j}+H_{j|i}}{2}$. 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.

Mode Pairs with Continuous Data. Let X_i and X_j be two continuous random variables; the conditional entropy, of X_i given X_j can be computed based on the formula:

$$H^*_{i|j} = H^*_{(i,j)} - H^*_j, \tag{13}$$

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

where $H^*_{(i,j)}$ is the joint entropy of X_i and X_j and H^*_j is the entropy estimate for mode j discussed in Section 5.3. Average pairwise entropy (ACE) can then be computed similarly as above.

Mixed Mode Pairs. Mode pairs could be mixed, e.g. (1) X_i is continuous and X_j is categorical *or* (2) X_i is categorical and X_j is continuous. Since there is no direct way to compute the joint entropy for mixed node pairs, here we approximate the joint entropy for X_i and X_j as $H^\star_{(i,j)}$ for later conditional entropy computation. And the approximation of the joint entropy for mixed node pairs is based on the property that $H_{(i,j)} <= H_i + H_j$.

Hence, the conditional entropy for mixed mode pair $H^\star_{i|j}$ could be estimated as:

- X_i is continuous and X_j is categorical:

$$H^\star_{i|j} \sim H^\star_{(i,j)} - H_j, \tag{14}$$

  where $H^\star_{(i,j)} \sim H^*_i + H_j$.

- X_i is categorical and X_j is continuous:

$$H^\star_{i|j} \sim H^\star_{(i,j)} - H^*_j, \tag{15}$$

  where $H^\star_{(i,j)} \sim H_i + H^*_j$.

In both cases, average pairwise entropy (ACE) can then be computed similarly as above.

Conditional Entropy Statistics for the Tensor. Given the above, we can then compute the average and standard statistics for ACE as follows:

$$\mu_{ace}(\mathcal{X}) = average(ACE_{(i,j)} \mid i \neq j), \tag{16}$$

$$\sigma_{ace}(\mathcal{X}) = stdev(ACE_{(i,j)} \mid i \neq j), \tag{17}$$

$$\phi_{ace}(\mathcal{X}) = \sigma_{ace}/\mu_{ace}. \tag{18}$$

The average and standard deviation statistics 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.

5.4. Pairwise Mutual Information

A related measure to conditional entropy is the pairwise mutual information.

Mode Pairs with Categorical Data. 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

$$\begin{aligned} MI_{(i,j)} &= \sum_{x \in X_i} \sum_{y \in X_j} p_{(X_i,X_j)}(x,y) \log\left(\frac{p_{(X_i,X_j)}(x,y)}{p_{X_i}(x) p_{X_j}(y)}\right) && (19) \\ &= H_i - H_{i|j} = H_j - H_{j|i}. && (20) \end{aligned}$$

where $p_{(X_i,X_j)}$ is the joint probability mass function of X_i and X_j.

Mode Pairs with Continuous Data. For mode pairs with *continuous* entries, we can leverage the entropy conditional entropy formulations presented in the previous subsections to compute pairwise mutual information:

$$MI^*_{(i,j)} = H^*_i - H^*_{i|j} = H^*_j - H^*_{j|i}. \quad (21)$$

Mixed Mode Pairs. If X_i is continuous and X_j is categorical, we approximate the pairwise mutual information $MI^\star_{(i,j)}$ as

$$MI^\star_{(i,j)} \sim H^*_i - H^\star_{i|j}. \quad (22)$$

If X_i is categorical and X_j is continuous, on the other hand, to avoid having to use an approximated value for $H^*_{i|j}$, we approximate the pairwise mutual information $MI^\star_{(i,j)}$ as

$$MI^\star_{(i,j)} \sim H^*_j - H^\star_{j|i}. \quad (23)$$

Pairwise Mutual Information Statistics for the Tensor. We then compute that average and standard statistics for mutual information as follows:

$$\begin{aligned} \mu_{mi}(\mathcal{X}) &= average(MI_{(i,j)} \mid i \neq j), && (24) \\ \sigma_{mi}(\mathcal{X}) &= stdev(MI_{(i,j)} \mid i \neq j), && (25) \\ \phi_{mi}(\mathcal{X}) &= \sigma_{mi}/\mu_{mi}. && (26) \end{aligned}$$

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.

6. GTT: Guiding Tensor Trains towards Highly Accurate Decompositions

Given the tensor data characteristics for discrete and continuous valued data described in the previous section, here we present various GTT strategies for guiding the tensor trains decomposition process.

6.1. GTT-NP: Number of Parameters

Consider the TT-decomposition process depicted in Figure 2. Here a 3-mode input tensor $\mathcal{X} \in \mathbb{R}^{l_{\pi_1} \times l_{\pi_2} \times l_{\pi_3}}$ is being converted into TT-format with a given decomposition sequence $\Pi = \langle \pi_1(mode_1), \pi_2(mode_2), \pi_3(mode_3) \rangle$ 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 $(r_{\pi_0} \times l_{\pi_1} \times r_{\pi_1}) + (r_{\pi_1} \times l_{\pi_2} \times l_{\pi_3}) + (r_{\pi_1} \times l_{\pi_2} \times r_{\pi_2}) + (r_{\pi_2} \times l_{\pi_3})$. It is easy to generalize this to

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

The first guiding strategy, 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.

6.2. GTT-AMI and GTT-PMI: Mutual Information

Aggregate Mutual Information (AMI). Mutual information (Equation 19 or Equation 23) 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.

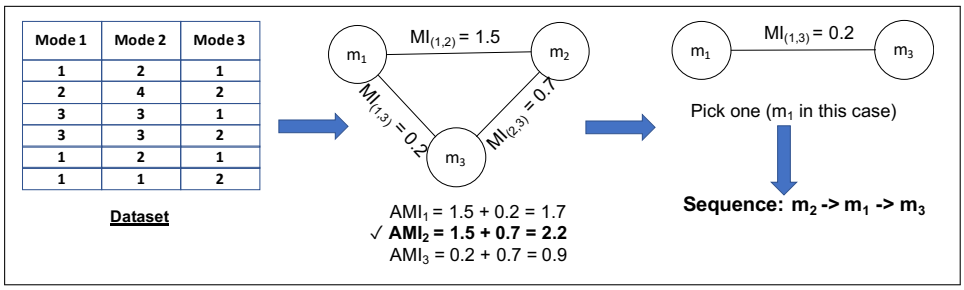

Figure 3: GTT-AMI computation for a 3-mode tensor. We first compute pair-wise mutual information and then aggregate these mutual information to decide which mode should be selected to decompose. In this example, Mode 2 will be first mode to decompose, since it has the largest aggregated mutual information.

More specifically, given a d-mode tensor, the AMI value for mode i is computed as

$$AMI_i = \sum_{j=1}^{d} MI_{(i,j)}.$$

We argue (and later experimentally show) that 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.

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).*

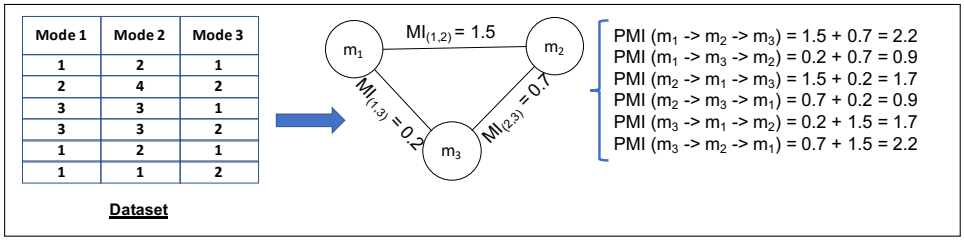

Figure 4: GTT-PMI computation for a 3-mode tensor. We first compute pair-wise mutual information and then for a given sequence of order (path), we accumulate its corresponding pair-wise mutual information and then select the sequence which has the lowest accumulated mutual information.

6.3. GTT-IE: (Inverse) Entropy

Remember that at the first step of the TT-decomposition process, we first matricize a 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 strong indicator of this is the entropy: if SV^T has high entropy, it is likely that it will

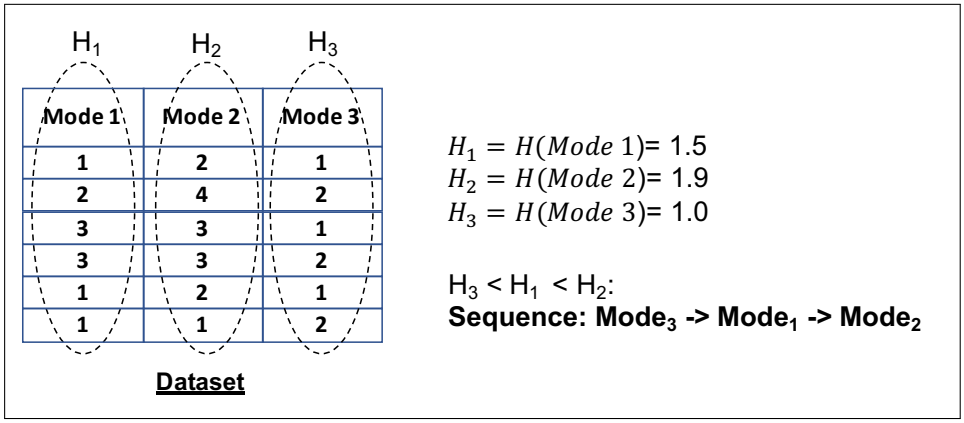

Figure 5: GTT-IE computation for a 3-mode tensor. We first compute entropy of each mode, and then decides a TT-decomposition sequence based on entropy in ascending order. In this example, the decomposition order is Mode 3 → Mode 1 → Mode 2.

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 with Equation 6 for each mode (H_1, H_2, H_3), and then decides a TT-decomposition sequence based on entropy in ascending order.

6.4. GTT-HYB: Hybrid Strategy

In Table 3, we list the data sets we use in our experiments 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.

Algorithm 2 GTT-HYB

Input:

• A d-mode tensor $\mathcal{X} \in \mathbb{R}^{l_{\pi_1} \times l_{\pi_2} \times \cdots \times l_{\pi_d}}$, where $\Pi = \langle \pi_1, \pi_2, \ldots, \pi_d \rangle$, is a permutation of modes in $\mathcal{X}$;

• A feature vector $X \in \mathbb{R}^m$ in Section 5 for $\mathcal{X}$;

• A list of target tt-ranks, $\langle r_{\pi_0}, r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_d} \rangle$, $r_{\pi_0} = r_{\pi_d} = 1$;

• Linear SVM classifiers for GTT-NP, GTT-PMI and GTT-IE:

f_{NP},f_{PMI}, f_{IE};

Output:

A sequence of decomposition order recommended by GTT-HYB.

• SG_X : Selected GTT for X.

• $f_{SG}(X, \langle r_{\pi_0}, r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_d} \rangle)$: Separation between X and the classifier for SG.

1: **procedure** GTT-HYB($X, \langle r_{\pi_0}, r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_d} \rangle$)

2: $SG_X = argmax_{SG \in \{NP,PMI,IE\}}(f_{SG}(X, \langle r_{\pi_0}, r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_d} \rangle))$.

3: return the sequence of decomposition order recommended by SG_X.

4: **end procedure**

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 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, with given training datasets (tensor instances)[5], for each scenario we consider the top-20% of the tensor instances for

[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 7, *GTT-PMI* is overall more accurate among the two, we omit *GTT-AMI* in hybrid selection.

[5]For GTT-HYB, we randomly sample 80% of tensor instances as training data, and the rest of 20% of tensor instances will be testing data. In experiments, we evaluate the proposed approaches with testing data.

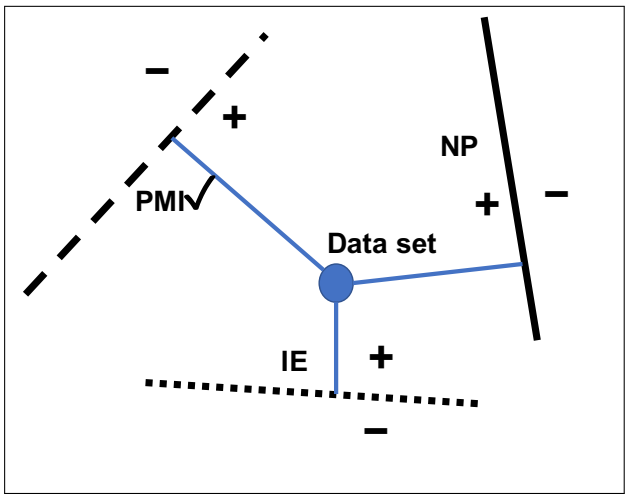

Figure 6: GTT-HYB selects a strategy using SVM classifiers. We train a linear SVM classifier for each GTT-strategy (GTT-NP, GTT-IE, and GTT-PMI), and then select a strategy which has the maximum separation for a given data set. In this example, PMI would be selected.

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. Figure 6 depicts the concept of our *GTT-HYB*: each GTT strategy has its linear SVM classifier. For a given data set. *GTT-HYB* selects the strategy which has the largest margin. In this example, *GTT-PMI* is selected since it has maximal separation. And the pseudo procedure of GTT-HYB is described in Algorithm 2.

6.5. 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 $O(t)$.

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[6], which is $O(dnr^3)$ and the number of parameters will be $O(dnr + (d-2)r^3)$.

7. Experimental Results

Here, we present experimental evaluations of the proposed GTT strategies[6]. Note that (once the decomposition order is selected) the data tensors are decomposed using TT-SVD [6] on a 4-core CPU (2.7GHz each) machine, with 16GB RAM. And further settings are in the following:

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

Table 2: Data sets [11]

Data set	**#Inst.**	**#Modes**	**Data set**	**#Inst.**	**#Modes**
Categorical-valued data					
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			
Continuous-valued data					
iris	5	16	abalone	237	14
wine	206	9			

Data Sets. We use both categorical and continuous valued data sets in our experiments. Table 2 lists the data sets we use in these experiments. The data sets are taken from the UCI Machine learning repository [11], where 15 of them are categorical-valued data sets and 3 of them are continuous-valued data sets. 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 (categorical-valued data) and 459 (continuous-valued data).

TT-Ranks. Here, we consider two TT-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 [27] which can learn also the appropriate rank, this is outside of the scope of this paper. We leave this to the future works.

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.

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.
Categorical-valued Data										
balance-scale	**0.73**	1.03	0.99	0.8	**0.73**	1.03	0.99	0.80	42%	43%
breast	**0.79**	1.08	1.02	0.99	**0.82**	1.09	0.99	0.97	6%	7%
car	**1.24**	0.90	1.02	0.69	1.26	1.23	1.03	**0.70**	-4%	1%
chess	**0.88**	0.97	1.01	0.89	**0.86**	0.97	1.01	0.88	-1%	-3%
dermatology	**0.85**	0.93	1.02	1.11	**0.88**	0.95	1.00	1.12	3%	4%
flare	0.92	0.89	**0.85**	1.18	0.94	0.92	**0.84**	1.17	4%	6%
hayes-roth	0.75	0.72	0.75	**0.67**	0.77	0.70	0.77	**0.69**	14%	7%
house_votes	0.96	1.03	**0.89**	1.18	0.93	1.04	**0.89**	1.16	-8%	-6%
lymphography	**0.73**	0.93	1.01	0.99	**0.74**	0.96	1.00	1.02	6%	5%
mushroom	0.92	0.87	**0.82**	1.10	1.02	**0.78**	0.86	1.12	3%	2%
nursery	0.90	0.76	0.92	**0.72**	0.94	**0.80**	0.93	**0.69**	10%	7%
primary-tumor	0.94	**0.68**	0.87	0.91	0.98	**0.62**	0.83	0.90	9%	12%
soybean	**0.84**	0.91	0.95	1.05	**0.87**	0.96	0.93	1.02	5%	3%
spect	0.97	1.00	0.92	**0.91**	0.97	1.00	0.91	**0.90**	8%	10%
tic-tac-toe	**0.70**	1.05	0.95	1.02	**0.70**	1.05	0.94	1.01	7%	17%
Average	**0.87**	0.92	0.93	0.95	**0.89**	0.94	0.93	0.94	7%	8%
*Inst. weighted Average	**0.86**	0.90	0.93	0.96	**0.88**	0.91	0.92	0.96	6%	7%

7.1. Evaluations for Categorical-valued Data Sets

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.

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	5.16	5.33	-	-	82.9	85.3
IE	4.90	5.07	4.9%	5.0%	**78.6**	**81.8**
NP	5.05	5.30	2.0%	0.7%	84.4	82.5
PMI	4.95	5.13	4.0%	3.9%	81.5	84.3
AMI	4.98	5.19	3.4%	2.7%	82.1	82.3
HYB	**4.86**	**5.01**	**5.8%**	**6.1%**	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	55.0	33.0	1.7	54.0	35.0	1.5
NP	36.0	25.0	1.4	35.0	26.0	1.4
PMI	47.0	31.0	1.5	47.0	32.0	1.5
AMI	45.0	42.0	1.1	45.0	42.0	1.1
HYB	50.0	26.0	**2.0**	51.0	24.0	**2.1**

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.

7.1.1. 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

house_votes 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 also shows that the hybrid strategy (*GTT-HYB*, described in Section 6.4) 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 12 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 6.4, 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 6.5, 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) 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	$\sigma_{ace}[3.2]$; $\phi_{ace}[-2.0]$; $\mu_{length}[1.8]$

Table 7: The relative average ranking against ARB (lower, the better) between *GTT-BEST* and *GTT-HYB* for each data set.

	TT-rank=3		**TT-rank-5**	
Dataset	**GTT-BEST**	**GTT-HYB**	**GTT-BEST**	**GTT-HYB**
balance-scale	0.73	0.73	0.73	0.99
breast	0.52	0.72	0.54	0.73
car	0.63	1.25	0.82	1.15
chess	0.59	0.95	0.59	0.92
dermatology	0.52	0.92	0.52	0.92
flare	0.47	0.71	0.45	0.74
hayes-roth	0.47	0.78	0.46	0.83
house_votes	0.56	0.96	0.53	0.89
lymphography	0.49	0.75	0.51	0.78
mushroom	0.47	0.74	0.48	0.75
nursery	0.42	0.86	0.48	0.94
primary-tumor	0.49	0.69	0.47	0.64
soybean	0.47	0.86	0.47	0.86
spect	0.78	0.97	0.77	0.96
tic-tac-toe	0.64	0.99	0.64	0.87
Average	0.55	0.86	0.56	0.86

- For *GTT-IE*, the two main contributors are σ_{ace} and ϕ_{ace}. This echos our argument in Section 6.3: *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 6.1, 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

Table 8: Relative average ranking against ARB selection strategy (lower, the better) for continuous-valued data.

tt-rank, r=3							
Statistics Collection	**Width**	**ARB**	**AMI**	**PMI**	**IE**	**NP**	**HYB**
DtS	5	1	1.19	1.42	0.88	1.29	1.04
	10	1	1.19	1.43	0.91	1.25	0.97
	15	1	1.16	1.44	1.05	1.24	0.99
StD	5	1	0.93	1.02	0.89	1.29	**0.82**
	10	1	0.90	1.04	0.92	1.28	**0.86**
	15	1	0.93	1.07	0.97	1.28	**0.84**
tt-rank, r=5							
Statistic Collection	**Width**	**ARB**	**AMI**	**PMI**	**IE**	**NP**	**HYB**
DtS	5	1	1.22	1.42	0.93	1.26	1.01
	10	1	1.17	1.42	0.96	1.24	0.97
	15	1	1.13	1.40	0.95	1.25	1.03
StD	5	1	1.01	1.07	0.85	1.28	**0.76**
	10	1	0.99	1.08	0.87	1.27	**0.75**
	15	1	0.99	1.06	0.90	1.24	**0.76**

Section 6.2: 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*.

Compare with the best strategy within GTT. To further analyze the performance of *GTT*, we compare the relative average ranking against ARB selection strategy between *GTT-BEST* (select the best result within *GTT-NP*, *GTT-PMI* and *GTT-IE*) and *GTT-HYB*. From the results in Table 7, we can see both *GTT-BEST* and *GTT-HYB* provide better results than ARB - the "average" decomposition performance of uninformed (i.e. arbitrary) order. However, *GTT-BEST* select decomposition order from the union set of GTT-NP, GTT-PMI, and GTT-IE, which means, this selection could be like enumerating all possible decomposition orders. Furthermore, *GTT-BEST* triples the computation effort and is not appropriate for the practical use, while *GTT-HYB* only needs one-time effort to train classifiers and then could be applied to most of

the data.

7.2. Evaluations for Continuous-valued Data Sets

Here we consider the continuous-valued data sets in Table 2 and evaluate the relative average ranking for each GTT strategy with various TT-ranks (3 and 5)[7] against ARB (lower, the better) with alternative statistics collection strategies, *discretization-then-statistics (DtS)* and *statistics-then-discretization (StD)*. For tensor encoding, we consider different discretization widths (5, 10, 15) for *equal-width* binning in Section 5.1.

As we see in the Table 8, we obtain the best overall results under the *StD* statistics collection strategy, using GTT-HYB approach for ordering tensor modes. This indicates that for continuous data, it is more effective to collect statistics in the continuous domain and, once the statistics are collected, then, the hybrid tensor-train guiding strategy is again the most effective approach. Below, we look at these results in further detail:

Statistics Collection Strategies. In Table 8, we can see the GTT strategies with *StD* have better performance than GTT strategies with *DtS* – as expected, this is especially true for entropy-based GTT strategies (*GTT-AMI*,*GTT-PMI*).

It is interesting that *GTT-IE* has overall better performance with *DtS* discretization scenario than *StD* scenario, we think it is because *GTT-IE* only focuses on the order of mode entropy individually instead of considering the relationships across modes like *GTT-AMI* and *GTT-PMI*. Hence, we believe *DtS* scenario simplifies the process of determining the order of model entropy in *GTT-IE* and results in better performance. However, *DtS* scenario diminishes the potential benefit from actual entropy for continuous-valued data, which degrades the performance of *GTT-AMI*, *GTT-PMI*, and also *GTT-HYB*.

[7] GTT-HYB is re-trained with continuous-valued data sets under the same settings in Section 6.4.

Discretization Width. As we see in Table 8, discretization width has only minimal effect on the results. Especially in *StD*, data characteristics are computed before the discretization process; therefore, the impact of the size of the discretization window is especially minimal.

TT-Ranks. As expected, using higher TT-rank generally provides better decomposition performance than lower TT-ranks – this is because higher TT-rank means we preserve more information during decomposition process. Beyond this, however, we do not see any significant impact of the TT-rank on the relative performance of GTT strategies or statistics collection strategies.

8. Conclusion

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 for both categorical and continuous data set. In particular, we proposed three order selection strategies, (a) *number of parameters (NP)*, (b) *aggregate mutual information (AMI, PMI)*, and (c) *inverse entropy (IE)* , for guiding the tensor train decomposition sequence and we 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 good decomposition sequences.

References

[1] J. D. Carroll, J.-J. Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition, Psychometrika 35 (3) (1970) 283–319. doi:10.1007/BF02310791.

[2] R. Harshman, Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis, UCLA Working Papers in Phonetics 16 (1970).

[3] L. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.

[4] T. Kolda, B. Bader, The TOPHITS model for higher-order web link analysis, in: Proceedings of Link Analysis, Counterterrorism and Security 2006, 2006.

[5] Y. Yamaguchi, K. Hayashi, Tensor decomposition with missing indices, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, AAAI Press, 2017, p. 3217–3223.

[6] I. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317. arXiv:https://doi.org/10.1137/090752286, doi:10.1137/090752286.

[7] A. Novikov, D. Podoprikhin, A. Osokin, D. Vetrov, Tensorizing neural networks, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, MIT Press, Cambridge, MA, USA, 2015, pp. 442–450.

[8] Y. Chen, X. Jin, B. Kang, J. Feng, S. Yan, Sharing residual units through collective tensor factorization to improve deep neural networks, in: IJCAI-18, IJCAI, 2018, pp. 635–641.

[9] C. Y. Ko, R. Lin, S. Li, N. Wong, Misc: Mixed strategies crowdsourcing, in: IJCAI-19, IJCAI, 2019, pp. 1394–1400. doi:10.24963/ijcai.2019/193.

[10] A. Novikov, M. Trofimov, I. Oseledets, Exponential Machines, arXiv e-prints (2016) arXiv:1605.03795arXiv:1605.03795.

[11] D. Dua, C. Graff, UCI machine learning repository (2017).

[12] Q. Zhao, G. Zhou, S. Xie, L. Zhang, A. Cichocki, Tensor ring decomposition, CoRR abs/1606.05535 (2016). arXiv:1606.05535.

[13] M. Li, K. S. Candan, M. L. Sapino, GTT: guiding the tensor train decomposition, in: Similarity Search and Applications - 13th International

Conference, SISAP 2020, Copenhagen, Denmark, September 30 - October 2, 2020, Proceedings, Vol. 12440 of Lecture Notes in Computer Science, Springer, 2020, pp. 187–202. doi:10.1007/978-3-030-60936-8_15. URL https://doi.org/10.1007/978-3-030-60936-8_15

[14] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500. doi:10.1137/07070111X.

[15] A. H. Phan, A. Cichocki, Parafac algorithms for large-scale problems, Neurocomputing 74 (11) (2011) 1970 – 1984, adaptive Incremental Learning in Neural Networks Learning Algorithm and Mathematic Modelling Selected papers from the International Conference on Neural Information Processing 2009 (ICONIP 2009).

[16] U. Kang, E. E. Papalexakis, A. Harpale, C. Faloutsos, Gigatensor: scaling tensor analysis up by 100 times - algorithms and discoveries, in: KDD, 2012.

[17] I. Jeon, E. E. Papalexakis, U. Kang, C. Faloutsos, Haten2: Billion-scale tensor decompositions, in: 2015 IEEE 31st ICDE, 2015, pp. 1047–1058.

[18] S. Huang, K. S. Candan, M. L. Sapino, Bicp: Block-incremental cp decomposition with update sensitive refinement, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16, ACM, New York, NY, USA, 2016, pp. 1221–1230. doi:10.1145/2983323.2983717.

[19] K. Batselier, W. Yu, L. Daniel, N. Wong, Computing low-rank approximations of large-scale matrices with the tensor network randomized svd, SIAM Journal on Matrix Analysis and Applications 39 (3) (2018) 1221–1244.

[20] L. Li, W. Yu, K. Batselier, Faster tensor train decomposition for sparse data, ArXiv (2019).

[21] O. Mickelin, S. Karaman, Tensor ring decomposition, CoRR abs/1807.02513 (2018). arXiv:1807.02513.

[22] K. Batselier, The trouble with tensor ring decompositions, CoRR abs/1811.03813 (2018). arXiv:1811.03813.

[23] R. Kohavi, G. H. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1) (1997) 273 – 324, relevance.

[24] L. Yu, H. Liu, Feature selection for high-dimensional data: A fast correlation-based filter solution, in: T. Fawcett, N. Mishra (Eds.), Proceedings, Twentieth International Conference on Machine Learning, Vol. 2, 2003, pp. 856–863.

[25] M. Dash, K. Choi, P. Scheuermann, Huan Liu, Feature selection for clustering - a filter solution, in: 2002 IEEE ICDM, 2002. Proceedings., 2002, pp. 115–122.

[26] M. Dash, H. Liu, J. Yao, Dimensionality reduction of unsupervised data, in: Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence, 1997, pp. 532–539. doi:10.1109/TAI.1997.632300.

[27] M. Imaizumi, T. Maehara, K. Hayashi, On tensor train rank minimization : Statistical efficiency and scalable algorithm, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 3930–3939.

[28] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd Edition, Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, Amsterdam, 2011.
URL http://www.sciencedirect.com/science/book/9780123814791

[29] C. E. Shannon, A mathematical theory of communication., Bell Syst. Tech. J. 27 (3) (1948) 379–423.
URL http://dblp.uni-trier.de/db/journals/bstj/bstj27.html#Shannon48

[30] E. T. Jaynes, Information theory and statistical mechanics, Phys. Rev. 106 (4) (1957) 620–630. doi:10.1103/PhysRev.106.620.
URL http://prola.aps.org/abstract/PR/v106/i4/p620_1

[31] L. F. Kozachenko, N. N. Leonenko, Sample estimate of the entropy of a random vector, Probl. Inf. Transm. 23 (1-2) (1987) 95–101.

[32] A. Kraskov, H. Stögbauer, P. Grassberger, Estimating mutual information, Phys. Rev. E 69 (2004) 066138. doi:10.1103/PhysRevE.69.066138.
URL https://link.aps.org/doi/10.1103/PhysRevE.69.066138