Multi-way block localization vis sparse tensor clustering

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

We consider the task of simultaneously clustering each mode of a large noisy tensor. We assume that the tensor elements are distributed with a block-specific mean and propose a least-square estimation for multi-way clustering. An ℓ_1 penalty is applied to the block-means in order to select and identify important blocks. We show that our method is applicable to large tensors with a wide range of multi-way cluster structure, including a single block, multiple blocks, checkerboard clusters, 1-way or lower-way blocks. Our proposal amounts to a sparse, multi-way version of k-mean clustering, and a relaxation of our proposal yields the tensor Tucker decomposition. The performance of our proposals are demonstrated in simulations and on...

1 Introduction

In recent years, much interest has centered around the unsupervised analysis of high-dimensional high-order tensor data.

Here is an example of tensor clustering by using our proposed method.

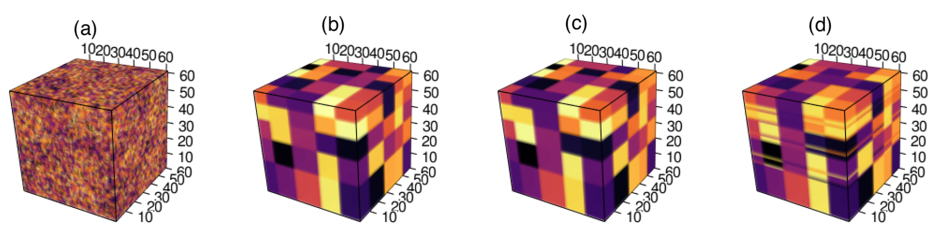


Figure 1: (a): a $60 \times 60 \times 60$ tensor with 5 clusters in each mode; (b): true underlying mean signal within each cluster; (c): mean signal estimated by our proposed approach with true number of clusters (5, 5, 5); (d): mean signal estimated by k-means clustering on each mode with true number of clusters (5 , 5, 5).

2 Preliminaries

We say that an event A occurs "with high probability" if $\mathbb{P}(\boldsymbol{A})$ tends to 1 as the dimension $d_{\min} = \min\{d_1, \ldots, d_k\}$ tends to infinity. We say that A occurs "with very high probability" if $\mathbb{P}(A)$ tends to 1 faster than any polynomial of d.

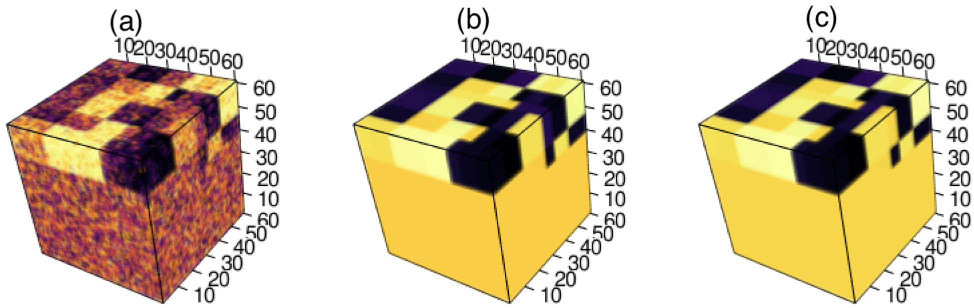


Figure 2: (a): $60 \times 60 \times 60$ sparse tensor; (b) true underlying means; (c) mean signal estimated by our approach with estimated number of clusters and estimated λ.

We use lower-case letters $(a, b, u, v, \dots)$ for scalars and vectors. We use upper-case boldface letters $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \dots)$ for matrices, and calligraphy letter $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \dots)$ for tensors of order $K \geq 3$. $\boldsymbol{x} \otimes \boldsymbol{y}$ is the Kronecker product of two vectors. For any set J, $|J|$ denotes its cardinality. $[d]$ represents the set $\{1, 2, \dots, d\}$.

A clustering of d objects can be represented by a partition of the index set $[d] = \{1, \dots, d\}$ into R disjoint non-empty subsets. We refer to R the clustering size. It is often convenient to represent the clustering (or partition) using the "membership matrix". A membership matrix $\boldsymbol{M}$ is an d-by-R matrix whose (i, j)-entry is 1 if and only if the element i belongs to the cluster j, and 0 otherwise. Based on the definition, $\boldsymbol{M}$ is binary matrix with orthogonal columns, and the matrix elements sum up to d. The membership matrix $\boldsymbol{M}$ can also be viewed as a mapping $\boldsymbol{M} : [d] \mapsto [R]$. With a little abuse of notation, we still use the $\boldsymbol{M}$ to denote the mapping, and use $\boldsymbol{M}(i)$ to denote the cluster label that entry i belongs to. Accordingly, we write $\boldsymbol{M} \in [R]^d$.

Throughout the paper, we will use the terms "partition", "clustering", and "membership matrix" exchangeably.

For a higher-order tensor, the above concepts can be applied to each of the modes. We use the term "mode-k clustering" to refer to the partition along the k-th mode of the tensor, and reserve "block" to mean a multi-way block in the Cartesian product of mode-k clusters.

3 Tensor block model

Let $\mathcal{Y} = [\![y_{i_1,\dots,i_K}]\!] \in \mathbb{R}^{d_1 \times \cdots \times d_K}$ denote an order-K, $(d_1, \dots, d_K)$-dimensional data tensor. The main assumption on tensor block model is that the observed data tensor $\mathcal{Y}$ is a noisy realization of an underlying tensor that exhibits a checkerbox structure (see Figure 1). Specifically, suppose that there are R_k clusters along the k-th mode of the tensor for $k \in [K]$. If the tensor entry $y_{i_1,\dots,i_K}$ belongs to the block jointly determined by the r_k-th mode-k cluster with $r_k \in [R_k]$, then we assume that

$$y_{i_1,\dots,i_K} = c_{r_1,\dots,r_K} + \varepsilon_{i_1,\dots,i_K}, \quad \text{for } (i_1, \dots, i_K) \in [d_1] \times \cdots \times [d_K], \tag{1}$$

where $\mu_{r_1,\dots,r_K}$ is the mean of the tensor block indexed by $(r_1, \dots, r_K)$, and $\varepsilon_{i_1,\dots,i_K}$'s are independent, mean-zero noise terms to be specified later. Our goal is to (i) find partitions along each of the modes, and (ii) estimate the block means $\{c_{r_1,\dots,r_K}\}$, such that a corresponding blockwise-constant checkerbox structure emerges in the data tensor.

The above tensor block model (1) falls into a larger class of non-overlapping, constant-mean clustering models [1], in that each tensor entry belongs to exactly one block with a common mean. The model (1) can be equivalently expressed as a special tensor Tucker model,

$$\mathcal{Y} = \mathcal{C} \times_1 \boldsymbol{M}_1 \times_2 \cdots \times_K \boldsymbol{M}_K + \mathcal{E}, \tag{2}$$

where $\mathcal{C} \in \mathbb{R}^{R_1 \times \cdots \times R_K}$ is a core tensor consisting of block means, $\boldsymbol{M}_k \in \{0, 1\}^{R_k \times d_k}$ are membership matrices indicating the block allocations along mode k for $k \in [K]$, and $\mathcal{E} = [\![\varepsilon_{i_1,\dots,i_K}]\!]$ is the noise tensor. The distinction between our model (2) and a classical Tucker model is that we require the factors $\boldsymbol{M}_k$ to be membership matrices. Our model (2) can be viewed as a super-sparse Tucker model, in the sense that the each column of $\boldsymbol{M}_k$ consists of one copy of 1's and massive 0's.

We now introduce the assumptions on the noise tensor $\mathcal{E}$. We assume that $\varepsilon_{i_1,\ldots,i_K}$'s are independent, mean-zero, σ-subgaussian noises, where $\sigma > 0$ is the subgaussianity parameter. More precisely,

$$\mathbb{E}e^{\lambda\varepsilon_{i_1,\ldots,i_K}} \leq e^{\lambda^2\sigma^2/2}, \quad \text{for all } (i_1,\ldots,i_K) \in [d_1]\times\cdots\times[d_K] \text{ and } \lambda \in \mathbb{R}. \tag{3}$$

Th assumption (3) is fairly general, which includes many common noises, such as Gaussian errors, Bernoulli errors, bounded errors, or even combinations of them. In particular, we consider two examples of the tensor block model that commonly appear in the literature:

Example 1 (Gaussian Multi-Cluster Model) *Let $\mathcal{Y}$ be a continuous-valued tensor. The Gaussian Multi-cluster model $y_{i_1,\ldots,i_K} \sim_{\text{i.i.d.}} N(\mu_{r_1,\ldots,r_K}, \sigma^2)$ is a special case of model* (1) *with the subgaussianity parameter σ equal to the error variance.*

Example 2 (Stochastic Block Model) *Let $\mathcal{Y}$ be a binary-valued tensor. The multiway stochastic block model $y_{i_1,\ldots,i_K} \sim_{\text{i.i.d.}}$ Bernoulli$(\mu_{r_1,\ldots,r_K})$ is a special case of model* (1) *with the subgaussianity parameter σ equal to $\frac{1}{4}$.*

More generally, our model also applied to hybrid error distributions in which different types of distribution can be allowed for different portions of the data. This scenario may happen, for example, when the data tensor $\mathcal{Y}$ represents concatenated measurements from multiple data sources.

We consider a least-square approach for estimating model (1). Let $\Theta = \mathcal{C}\times_1 \boldsymbol{M}_1 \times_2 \cdots \times_K \boldsymbol{M}_K$ denote the mean signal tensor with block structure. The mean tensor is assumed to belong to the following parameter space

$$\mathcal{P}_{R_1,\ldots,R_K} = \big\{\Theta \in \mathbb{R}^{d_1\times\cdots\times d_K} : \Theta = \mathcal{C}\times_1 \boldsymbol{M}_1 \times_2 \cdots \times_K \boldsymbol{M}_K, \text{with some} \tag{4}$$

$$\text{membership matrices } \boldsymbol{M}_k\text{'s and a core tensor } \mathcal{C} \in \mathbb{R}^{R_1\times\cdots\times R_K}\big\}. \tag{5}$$

We note that the clustering sizes $(R_1,\ldots,R_K)$ is typically unknown and have to be determined from data empirically. As in most previous work on tensor clustering, we assume that clustering sizes are fixed in our theoretical analysis and simply write $\mathcal{P}$ for short. The general case for adapting unknown clustering sizes will be addressed in Section 5.2. We propose a least-square estimator for model (1)

$$\hat{\Theta} = \arg\min_{\Theta\in\mathcal{P}} \left\{-2\langle \mathcal{Y}, \Theta\rangle + \|\Theta\|_F^2\right\}. \tag{6}$$

The objective is equal (ignoring constants) to the sum of squares $\|\mathcal{Y}-\Theta\|_F^2$ and hence the name of our estimator. Estimating Θ consists of finding both the core tensor $\mathcal{C}$ and the membership matrix estimates $\boldsymbol{M}_k$'s. Before we discuss the properties of $\hat{\Theta}$, we present the identifiability of $\boldsymbol{M}_k$'s and $\mathcal{C}$ from Θ.

The following irreducible assumption is necessary for the tensor block model to be identifiable.

Assumption 1 (Irreducible cores) *The core tensor $\mathcal{C}$ is called irreducible if it cannot be written as a block tensor with the number of mode-k clusters smaller than R_k, for any $k \in [K]$.*

In the matrix case $(K = 2)$, the assumption is equivalent to saying that $\mathcal{C}$ has no two identical rows and no two identical columns. In the higher-order case, it requires that none of order-(K-1) fibers of $\mathcal{C}$ are identical. Note that the being irreducible is a weaker assumption than being full rank.

Proposition 1 (Identifiability) *Consider a Gaussian or Bernoulli tensor block model* (2)*. Suppose the core tensor satisfies Assumption 1. Then every factor matrix $\boldsymbol{M}_k$ is identifiable up to permutations of cluster labels.*

Our identifiability result is stronger than the classical Tucker model. In a classical Tucker model [2, 3] and many other factor analyses [4, 5], the factors are identifiable only up to orthogonal rotations. In those models, the (column) space spanned by $\boldsymbol{M}_k$ can be recovered, but not the individual factors. In contrast, our model does not suffer from rotational invariance, and as we show in Section 4, every single factor can be consistently estimated in high dimensions. This brings a benefit to the interpretation of tensor factors in the block model.

4 Statistical convergence

In this section, we assess the estimation accuracy of the least-squares estimator (6). While the least squares corresponds to the maximum likelihood estimator (MLE) for Gaussian tensor model, the same assertion does not hold for the other types of distribution such as stochastic tensor block model. Surprisingly, we will show that, with very high probability, a simple least-square estimator can achieve a convergence rate that is nearly the same as the optimal one in a general class of block tensors.

For the true signal tensor $\Theta_{\text{true}} \in \mathcal{P}$ and its estimator $\hat{\Theta} \in \mathcal{P}$, define

$$\text{Loss}(\Theta_{\text{true}}, \hat{\Theta}) = \frac{1}{\prod_k d_k} \|\Theta_{\text{true}} - \hat{\Theta}\|_F^2.$$

Theorem 1 (Convergence) *Let $\hat{\Theta}$ be the least-square estimator of $\Theta_{\textit{true}}$ under model* (1). *There exist two constants $C_1, C_2 > 0$ such that, with very high probability,*

$$\textit{Loss}(\Theta_{\textit{true}}, \hat{\Theta}) \leq \frac{C_1 \sigma^2}{\prod_k d_k} \left(\prod_k R_k + \sum_k d_k \log R_k \right), \tag{7}$$

holds uniformly over $\Theta_{\textit{true}} \in \mathcal{P}_{\boldsymbol{R}}$ and all error distribution satisfying (3).

The convergence rate in (7) consists of two parts. The first part $\prod_k R_k$ reflects the complexity for estimating the core tensor $\mathcal{C}$, while the second part $\sum_k d_k \log R_k$ results from the complexity for estimating the supports of $\boldsymbol{M}_k$'s. It is the price that one has to pay for not knowing the locations of the blocks.

We now compare our bound with existing literature. The classical Tucker tensor decomposition has a minimax convergence rate $\sum_k d_k R'_k$ [2], where R'_k is the multilinear rank at mode k. In the case of block model, this yields $\sum_k d_k R_k$, because the mode-k rank is bounded by the number of clusters in mode-k. Now, as both the dimension $d_{\min} = \min_k d_k$ and clustering size $R_{\min} = \min_k R_k$ tend to infinity, we have $\prod_k R_k + \sum_k d_k \log R_k \ll \sum_k d_k R_k$. Therefore, by fully exploiting the block structure, we obtain a better convergence rate than previously possible.

Our bound also generalizes the previous results on structured matrix estimation in network analysis [6, 7]. The optimal convergence rate for estimating the (matrix) stochastic block model was $R_1 R_2 + d_1 \log R_1 + d_2 \log R_2$ [6], which fits into our special case when $K = 2$. Earlier work [7] suggests the following heuristics on the sample complexity for high-dimensional matrix problems:

$$\frac{(\text{number of parameters}) + \log(\text{complexity of models})}{\text{number of samples}}. \tag{8}$$

Our result supports this important principle for general $K \geq 2$. Note that, in tensor estimation, the total number of entries corresponds to the sample size $\prod_k d_k$, the number of parameters is $\prod_k R_k$, and combinatoric complexity for estimating block models is of order $\prod_k R_k^{d_k}$. The principle (8) thus provide an intuition for (7).

The optimality of the least-square estimator is safeguarded by the following information-theoretical lower bound.

Theorem 2 (Minimax) *Under the Gaussian or Bernoulli tensor block models, there exist some constants $\alpha_0 > 0, \beta_0 \in (0, 1)$, such that*

$$\inf_{\hat{\Theta}} \sup_{\Theta \in \mathcal{P}} \mathbb{P} \left\{ \|\hat{\Theta} - \Theta_{\textit{true}}\|_F^2 > c_0 \sigma^2 \prod_k R_k + \sum d_k \log R_k \right\} > \beta_0.$$

(add the proof)

We next study the clustering consistency of our method. Let $\hat{\boldsymbol{M}}_k$ denote the estimated membership matrix corresponding to $\hat{\Theta}$, and $\boldsymbol{M}_{k,\text{true}}$ the true membership matrix. We define the clustering error rate as $\text{CER}(\hat{\boldsymbol{M}}_k, \boldsymbol{M}_{k,\text{true}}) = d_k^{-1} \sum_{i \in [d_k]} \mathbb{1}_{\{\hat{\boldsymbol{M}}_k(i) = \boldsymbol{M}_{k,\text{true}}(i)\}}$.

The following Theorem implies that our method achieves clustering consistence.

Theorem 3 (Clustering consistency) *Suppose the Assumption* (1) *holds. Let $\hat{M}_k$'s be the estimators from* (6). *Then the proportions of misclassified indices goes to zero in probability; i.e. there exist permutation matrices P_k's such that*

$$\sum_k CER(\hat{M}_k, P_k M_{k,true}) \to 0, \quad \textit{in probability}.$$

(add the proof) Under stronger distribution assumptions on $\mathcal{E}$, we can establish the finite-sample convergence rate for the clustering accuracy. See more results in the Supplements.

5 Numerical Implementation

5.1 Alternating optimization

We introduce an alternating optimization for solving (6). Note that the optimization (6) can be written as

$$(\hat{\mathcal{C}}, \{\hat{M}_k\}) = \mathop{\arg\min}_{\mathcal{C}\in\mathbb{R}^{R_1\times\cdots\times R_K},\ \text{membership matrices } M_k\text{'s}} f(\mathcal{C}, \{M_k\}),$$
$$\text{where} \quad f(\mathcal{C}, \{M_k\}) = \|\mathcal{Y} - \mathcal{C}\times_1 M_1 \times_2 \ldots \times_K M_K\|_F^2.$$

The decision variables consists of $K+1$ blocks of variables, one for the core tensor $\mathcal{C}$ and K for the membership matrices M_k. Given the collection of $\hat{M}_k$'s, the core tensor estimate $\hat{\mathcal{C}} = \arg\min_{\mathcal{C}} f(\mathcal{C}, \{\hat{M}_k\})$ has a closed-form solution, which is simply the sample average within each tensor block. Given $\hat{\mathcal{C}}$ and the $K-1$ membership matrices, the estimation of the mode-k clustering reduces to a classical k-mean problem. This observation suggests that we can iteratively update one block of variables at a time while keeping other others fixed. The full procedure is described in Algorithm 1.

Algorithm 1 Multiway clustering for tensor block models

Initialize the marginal clusterings by performing one-way k-means clustering on each of the modes on the tensor $\mathcal{Y}$.
repeat
(a) Update the core tensor $\mathcal{C}$:
Holding the clustering fixed, minimize the objective with respect to $\mathcal{C}$. for each $(r_1, \ldots, r_K) \in [R_1]\times\cdots[R_K]$,

$$C_{r_1,\ldots,r_K} = \frac{1}{\prod_k |M_k^{-1}(i_k)|} \sum_{M_1^{-1}(r_1)\times\cdots\times M_K^{-1}(r_K)} \mathcal{Y}_{i_1,\ldots,i_K}$$

(b) Update membership matrices M_k's:
for k in $\{1,2,...,K\}$ **do**
Minimize the objective with respect to M_k by assigning each mode-k index to the cluster whose mean signal is most closet to it. Update M_k: for each $a \in [d_k]$,

$$M_k(a) = \mathop{\arg\min}_{r\in[R_K]} \sum_{I_{-k}} \left(\mathcal{C}_{M_1(i_1),\ldots,r,\ldots,M_K(i_K)} - \mathcal{Y}_{i_1,\ldots,a,\ldots,i_K}\right)^2$$

where $I_{-k} = (i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_K)$ denotes the indices in all modes except the k-th mode.
end for
until Convergence

The above algorithm can be viewed as a higher-order K-means clustering in which the block means $\mathcal{C}$ serve as the role of centroids. Each iteration will reduces the value of the objective function, which is bounded from above. Therefore the convergence is guaranteed. Since the objective is non-convex, one cannot guarantee a local optimum. Following , we initialize the algorithm multiple times using random initialization.

5.2 Tuning Parameter Selection

Before doing tensor clustering, we need to select appropriate tuning parameters. There are $K+1$ tuning parameters in our tensor clustering proposal: the number of clusters in each modes: d_1, d_2, ..., d_K and the penalty coefficient λ. For both the number of clusters and the penalty coefficient, we try to use BIC and cross validation to find the best choice. It turns out the BIC is faster when the accuracy is almost the same, so we use BIC to select the tuning parameters.

$$\text{BIC}(\lambda, \boldsymbol{R}) = \log\left(\|\mathcal{Y} - \hat{\Theta}\|_F\right) + \frac{\sum_k \log d_k}{\prod_k d_k} p_e,$$

where p_e is the effective number of parameters. We minimize BIC by a grid search. First select $\boldsymbol{R}$ given $\lambda = 0$, then update λ using $\boldsymbol{R}$.

As for the number of clusters, given a range of d_1, d_2, ..., d_K, we do the tensor clustering for all combinations of d_1, d_2, ..., d_K with $\lambda = 0$ and calculate the BIC for each of them separately using the formula above. We choose the d_1, d_2, ..., d_K which is the smallest among all combinations of d_1, d_2, ..., d_K whose BIC is the smallest.

After estimating the d_1, d_2, ..., d_K, we use the estimated number of clusters to do tensor clustering when given a reasonable range of λ. We perform the tensor clustering and calculate the BIC on all λ in the given range. Then we select the smallest λ with smallest BIC.

6 Regularized estimation

In practice, the data tensor few blocks plus white noisy. See (Figure). There are a large number of regularization techniques for different purpose. Here we illustrate with using *sparsity* regularization on block means for localizing the most important blocks among tensor entries. This problem can be viewed as an analogue of variable selection but now the block serves the role of variables. We propose the following regularized least square

$$\underset{\Theta\in\mathcal{P}}{\arg\min}\left\{\|\mathcal{Y}-\Theta\|_F^2 + \lambda\|\mathcal{C}\|_\rho\right\},$$

where $\|\mathcal{C}\|_\rho$ is the penalty function, λ is the penalty tuning parameter, and ρ is an index for the tensor norm. Because our goal is to penalty the entries with small mean, the Lasso penalty $(\rho = 1)$ or sparse sub-set penalty $(\rho = 0)$ is suitable.

Sparse estimation incurs slight changes to Algorithm. When updating the core tensor $\mathcal{C}$, we simply fit a penalized least square problem..

$$\hat{c}_{\text{sparse}} = \begin{cases} \hat{c}_{\text{ols}} \mathbb{1}_{\{|\hat{c}_{\text{ols}}| \geq \frac{\lambda}{\sqrt{n}}\}} & \rho = 1, \\ \text{sign}(\hat{c}_{\text{ols}})\left(\hat{c}_{\text{ols}} - \frac{\lambda}{n}\right) & \rho = 0. \end{cases}$$

(See Lemma in Appendix). We choose to penalize the block means $\mathcal{C}$ but not the signal tensor Θ. (do not want to penalize blocks of small size but with large entry values). Applying penalization to Θ amounts to choose different penalization parameters to $c_{r_1,\ldots,r_K}$.

7 Simulation and Evaluation

For simplicity, we only consider the situation $K = 3$ here. Given the approach of clustering in the former section, we will evaluate the performance in different aspects on non-sparse and sparse tensor. For non-sparse tensor, we assess the following four aspects.

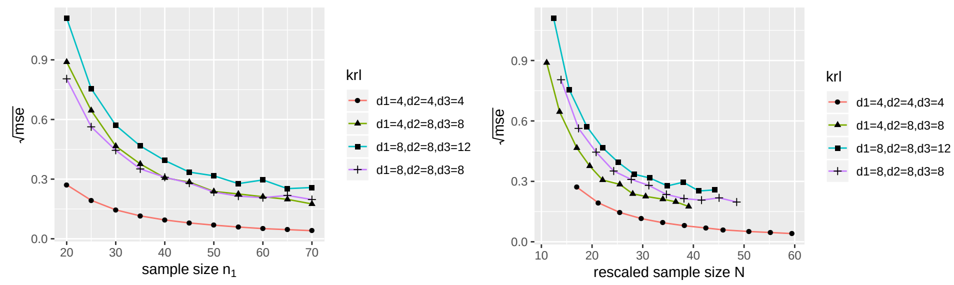
first, we assess the relationship between MSE and the data size;

second, we verify the clustering approach when true d_1, ..., d_K is given;

third, we evaluate the accuracy for estimating the number of clusters;

fourth, we evaluate the synergistic performance of selecting d_1, ..., d_K and clustering;

fifth, we would check the performance of our approach in a different kind of tensor and compare it with other clustering approach.



(a) (b)

Figure 3: Plots of the root mean squared error (RMSE) verus sample size when using our tensor clustering algorithm. Each curve corresponds to a fixed (d_1, d_2, d_3). (a): Plots of average RMSE over 50 simulations against n_1; (b): Plots of averge RMSE over 50 simulations against $\sqrt{n_2 n_3 / log d_1}$.

As for sparse tensor, we would evaluate the whole process: selecting d_1, ..., d_K, choosing λ, doing tensor clustering.

There are the statistics we would use to evaluate the performance in different cases:
(1) CER (clustering error rate): the adjusted rand index between two partitions. This statistic measures the agreement between the true partition and estimated partition of the data tensor. In this case, we have three kinds of CER in total: CER of mode 1, CER of mode 2 and CER of mode 3;
(2) MSE: measuring the difference between the underlying means and the estimated means.

For sparsity tensor, we further consider the following three statistics: (3) Total Correct Rate: 1 - the proportion of misjudgment while determining whether the mean signal is zero.;
(4) Correct Zero Rate: the proportion of zero elements are correctly identified in the underlying mean tensor;
(5) Correct One Rate: the proportion of non-zero elements are correctly identified in the underlying mean tensor.

In non-sparse cases, we mainly use CER (both CER and MSE??) as an indicator to assess our proposal methods because it is more intuitive for us to evaluate the performance of our approach comparing with MSE.

However, it has some constraints. In sparse cases, different clusters can have the same mean: 0. In this case, we can have multiple reasonable partitions of the modes. Thus, CER is inapplicable at this time and we use total correct rate, correct zero rate and correct one rate to be the indicator when the tensor is sparse.

Here we give a brief elaboration on the main way we use to generate the data. As for non-sparse tensor, given the cluster numbers d_1, ..., d_K and the size of the tensor $n_1 n_2 n_3$, we assign the labels to each modes randomly. Next we randomly select the mean signal of clusters from Unif(-3,3) and add noise which comes from normal distribution with given standard deviation. Then we get the non-sparse tensor. As for sparse tensor, we randomly assign 0 to the mean of some clusters with given sparsity rate (the proportion of 0 elements) and then follow the same steps. We name this kind of tensor as tensor with constant clusters.

Non-sparse case. We begin with verifying the relationship between MSE and the sample size. The theoretical result indicates that the boundary of $RMSE = \sqrt{MSE} = \sqrt{\frac{error}{n_1 n_2 n_3}}$ decreases with respect to sample size. Here we let n_1 to take values from 20 to 70, and $n_2 = \frac{n_1 log d_1}{log d_2}$,, $n_3 = \frac{n_1 log d_1}{log d_3}$. As for d_1, d_2, d_3, we take them from $\{(4,4,4), (4,8,8), (8,8,8), (8,8,12)\}$. We simulated each situation 50 times and get the average RMSE among those cases. According to the panel (a) of Figure 3, obviously, with sample size going up, the RMSE goes down. Additionally, the panel (b) of Figure 3 indicates the RMSE decreases roughly at the rate of $1/N$ where $N = \sqrt{n_2 n_3 / log d_1}$ is the rescaled sample size.

n_1	n_2	n_3	noise	CER(mode 1)	CER(mode 2)	CER(mode3)
40	40	40	4	$\mathbf{0}(\mathbf{0})$	$\mathbf{0}(\mathbf{0})$	$\mathbf{0}(\mathbf{0})$
40	40	40	8	$\mathbf{0}(\mathbf{0})$	0.0136(0.0226)	0.0005(0.0036)
40	40	40	12	0.0365(0.0789)	0.12(0.0878)	0.0802(0.1009)
40	45	50	4	$\mathbf{0}(\mathbf{0})$	$\mathbf{0}(\mathbf{0})$	$\mathbf{0}(\mathbf{0})$
40	45	50	8	$\mathbf{0}(\mathbf{0})$	0.0027(0.0121)	$\mathbf{0}(\mathbf{0})$
40	45	50	12	0.0158(0.0489)	0.0641(0.0629)	0.0336(0.0647)

Table 1: The CERs over 50 simulated tensors ($d_1 = 3, d_2 = 5, d_3 = 4$) each time.

In the second simulation, we generate 50 non-sparse tensors with the same noise, size and cluster numbers each time. We use our approach (Algorithm 1) to do the clustering and the result is shown as Table 1. In both data size: 40*40*40 and 40*40*80, the CER on all modes are 0 when the noise is 4. As the noise goes up, the CER is increased gradually. Furthermore, from the $d_1 = 3, d_2 = 5, d_3 = 4$ cases, we notice that the CER of mode i seems to be smaller when the number of clusters in the ith mode is less.

To evaluate the performance of our approach on selecting the number of clusters, we generate 50 non-sparse tensors with the same noise, size and cluster numbers in each case in Table 2 as the third simulation. The reason why we only evaluate the performance of estimation on cluster numbers on non-sparse tensor is in sparse case, the reasonable cluster numbers may not be unique. As expected, we achieve 100% accuracy when the noise is 4, again. The overall accuracy goes down as the noise increased. Additionally, we notice that the smaller d_i is, the more accurate the estimated value is. There are two extremely low overall accuracy which appears when noise is 12 and the tensor size is 40*40*40. However, the accuracy is improved quickly as the tensor size is enlarged to 40*40*80. We found that not only the accuracy of mode 3 decreased after we add observations on mode 3, but also the accuracy of other two modes decreased a lot. The reason is the length of an observation in mode 1 and mode 2 is longer than before. Therefore, it is very important for us to get enough observations to guarantee the overall accuracy.

In the forth simulation, the true cluster numbers are not given, so we estimate them first and then use the estimated true cluster numbers to estimate the partition of clusters as well as underlying mean signals. We set the true cluster numbers to be $d_1 = 3, d_2 = 5, d_3 = 4$ specifically here, and the results are shown in Table 1. By looking into each mode separately, as the sample size of that mode increased , the CER of that mode decreased without any exception.

In the fifth simulation, we generate the tensor in two different ways. The new way to generate data is using $\sum_{s=1}^{S} d_s \mathbf{u_s} \mathbf{v_s}^T$ after giving the S. First we randomly choose $\mathbf{u_s}$ and $\mathbf{v_s}$ from normal distribution where $i = 1, ..., S$. Then we add noise which is also sampled from normal distribution with given standard deviation. We name the tensor generated in this way as tensor with multiplicative clusters. In this simulation, we would check whether our method is still robust on the tensors with multiplicative cluster. Meanwhile, we would compare the performance of our approach with CPD k-means (do k-means clustering after using CP decomposition on the data tensor) under different noise and different kinds of tensor. To be more specific, when the data a a tensor with multiplicative clusters we just use the true rank that we use to generate the data tensor to do the CP k-means. As for the parameter d_1, d_2, d_3 in k-means, we also use the true parameter. We apply the true d_1, d_2, d_3 in our method, too. When the data is a tensor with constant clusters, we still use true d_1, d_2, d_3 in both methods. But for CPD k-means, we use BIC to select the best S according to the formula: $error + \frac{log(n_1 n_2 n_3)(n_1+n_2+n_3-2)S}{n_1 n_2 n_3}$. Here we set $n_1 = 50, n_2 = 50, n_3 = 50$ and $d_1 = 4, d_2 = 4, d_3 = 4$ in all the cases, and add noise 0,10,20 to generate the tensors. Additionally, as for tensor with constant clusters, I randomly select underlying signals between -3 and 3; as for tensors with multiplicative clusters, I randomly select the elements of $\mathbf{u_s}$ and $\mathbf{v_s}$ between -3 and 3. Thus the mean signal of tensor with multiplicative clusters would be bigger than that of tensors with constant clusters. Therefore, the same noise would have bigger effect on the tensor with constant clusters. Our result is shown in Figure 4. To our surprise, our approach is not only better at working on tensor with constant clusters, but also better at working on tensor with multiplicative clusters. Furthermore, our approach shows more robustness under different level of noise, too.

Sparse case. We also test the performance of our approach under different λ when the data is sparse. The $\bar{\lambda}$ in Table 3 is the mean λ we choose across 50 simulations on the same sparsity rate. According

(a) (b)

Figure 4: Plots of the mean CER C verus noise when using our tensor clustering algorithm and CPD k-means where $n_1 = n_2 = n_3 = 50, d_1 = d_2 = d_3 = 4$. Each curve corresponds to different method. (a): Plots of average CER in mode 1 over 50 simulations against noise while the data is a tensor with multiplicative clusters where $S = 3$; (b): Plots of averge CER in mode 1 over 50 simulations against noise while the data is a tensor with constant clusters.

to Table 3, the correct zero rate is increased with the increment on λ while the correct one rate is exactly the opposite. As for the λ we choose, it shows that the lowest total correct rate is 0.8586, which appears when noise is 8 and sparsity rate is 0.8. Overall, the λ we select by BIC is fairly good, which does works better than other λ.

8 Conclusion

Sparsity is only one form of regularization. In specific applications, prior knowledge often suggests various constraints among parameters, which may be exploited to regularize parameter estimates. For example, in the stochastic block model, sometimes it may be reasonable to impose symmetry on the parameters along certain subsets of modes, which further reduces the dimension of the problem. In some other applications, non-negativity of parameter of parameter values may be enforced. In our software, we implement the common penalizations but leave .. .to further study.

References

[1] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):24–45, 2004.

[2] Anru Zhang and Dong Xia. Tensor SVD: Statistical and computational limits. *IEEE Transactions on Information Theory*, 2018.

[3] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[4] Robin A Darton. Rotation in factor analysis. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 29(3):167–194, 1980.

[5] Hervé Abdi. Factor rotations in factor analyses.

[6] Chao Gao, Yu Lu, Zongming Ma, and Harrison H Zhou. Optimal estimation and completion of matrices with biclustering structures. *The Journal of Machine Learning Research*, 17(1):5602–5630, 2016.

[7] Chao Gao and Zongming Ma. Minimax rates in network analysis: Graphon estimation, community detection and hypothesis testing. *arXiv preprint arXiv:1811.06055*, 2018.