Multi-way block localization vis sparse tensor clustering

Yuchen Zeng
University of Wisconsin – Madison
yzeng58@wisc.edu

Miaoyan Wang
University of Wisconsin – Madison
miaoyan.wang@wisc.edu

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.

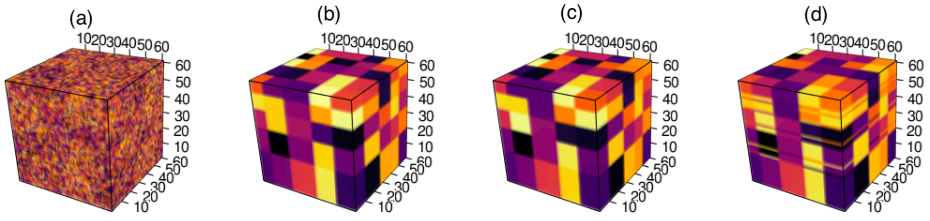


Figure 1: (a): a 60*60*60 tensor with 5 clusters in each mode; (b): true underlying mean signal within each cluster; (c): mean signal estimated by our proposed approach with true number of clusters: 5, 5, 5; (d): mean signal estimated by k-means clustering on each mode with true number of clusters:5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

1.1 Notation

We use $\mathcal{T}$, $\mathcal{X}$, and $\mathcal{E}$ to represent input, signal, and noise tensors, respectively. For any set J, $|J|$ denotes its cardinality. $[n]$ represents the set $\{1, 2, \ldots, n\}$. $\boldsymbol{x} \otimes \boldsymbol{y}$ is the Kronecker product of two vectors.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

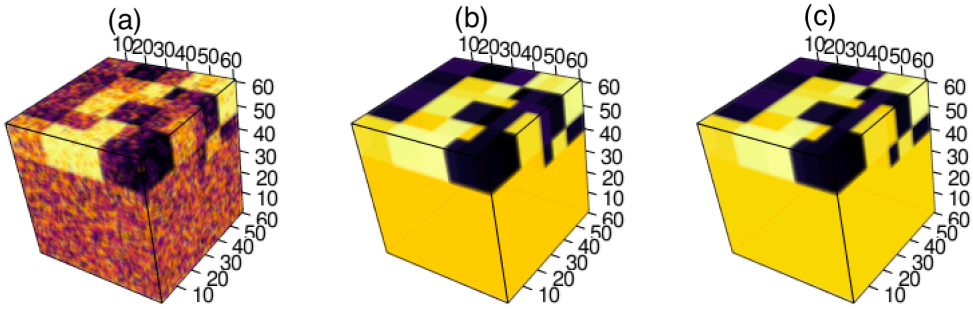


Figure 2: (a): 60*60*60 sparse tensor; (b) true underlying means; (c) mean signal estimated by our approach with estimated number of clusters and estimated λ.

1.2 Problem formulation

Notations:
1. K: the number of dimensions of the data;
2. n_i: the number of observations in the ith mode, $i = 1, 2, ..., K$;
3. d_i: the number of clusters in the ith mode, $i = 1, 2, ..., K$;
4. c_{ij}: the jth cluster of the ith mode, $i = 1, 2, ..., K$, $j = 1, ..., d_i$;
5. q: the number of non-zero clusters;
6. c: the degree of freedom while doing tensor clustering: $\sum_{i=1}^{K} d_i log(n_i)$.

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.

$$BIC = \prod_{i=1}^{K} n_i \times log(RSS) + (q + c) log(\prod_{i=1}^{K} n_i)$$

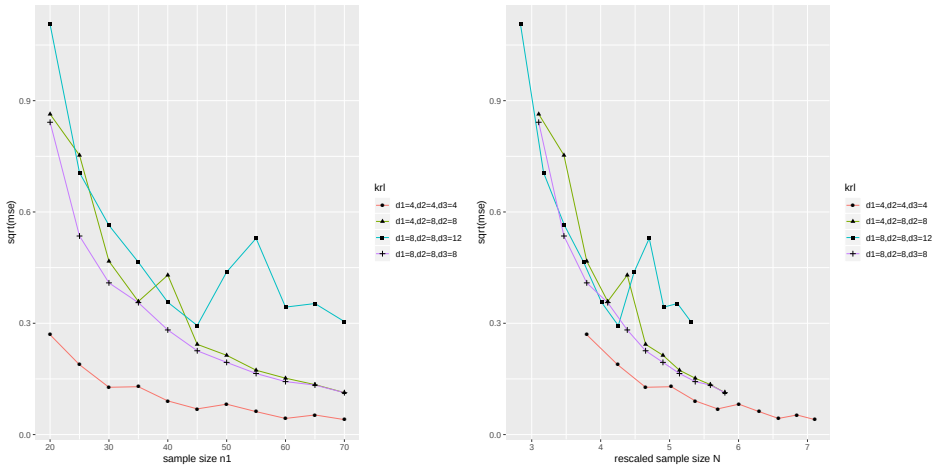
As for the number of clusters, given a range of d_1, d_2, ..., d_K, we do the tensor clustering for all combinations of d_1, d_2, ..., d_K with $\lambda = 0$ and calculate the BIC for each of them separately using the formula above. We choose the d_1, d_2, ..., d_K which is the smallest among all combinations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

3 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 of it in different aspects on non-sparse and sparse tensor. When the tensor is non-sparse, first, we would assess the relationship between MSE and the data size; second, we would verify the clustering approach when true d_1, ..., d_K is given; third, we would evaluate the approach of estimating the number of clusters; fourth, we would evaluate the synergistic performance of selecting d_1, ..., d_K and clustering. 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 paritions. This statistic measure



(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 RMSE against n_1; (b): Plots of RMSE against $\sqrt{n_1/logk_3}$.

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) Total Correct Rate: 1 - the proportion of misjudgement while determining whether the mean signal is zero.;
(3) Correct Zero Rate: the proportion of zero elements are correctly identified in the underlying mean tensor;
(4) Correct One Rate: the proportion of non-zero elements are correctly identified in the underlying mean tensor.

In non-sparse cases, we use CER as an indicator to judge whether our proposal methods are good or bad. However, in sparse cases, we use total correct rate, correct zero rate and correct one rate to be the indicator. This is because when the data tensor is sparse, then 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.

Here we give a brief elaboration on how to generate the data. As for non-sparse tensor, given the cluster numbers d_1, ..., d_K and the size of the tensor $n_1n_2n_3$, 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.

Non-sparse case. We begin with verifying the relationship between MSE and the sample size. The theoretical result indicates that the boundary of $RMSE = \sqrt{MSE} = \sqrt{\frac{error}{n_1n_2n_3}}$ decreases with respect to sample size. Here we let n_1 to take values from 20 to 70, and $n_2 = \frac{n_1logd_1}{logd_2}$,, $n_3 = \frac{n_1logd_1}{logd_3}$. As for d_1, d_2, d_3, we take them from $\{(4,4,4),(4,8,8),(8,8,8),(8,8,12)\}$. 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 $1/\sqrt{n_1/logk_3}$ where $N = \sqrt{n_1/logk_3}$ is the rescaled sample size.

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*45*50, the CER on all modes are 0 when the noise is 4. As the noise goes up, the CER is increased gradually. Furthemore, from the $d_1 = 3, d_2 = 5, d_3 = 4$

n_1	n_2	n_3	d_1	d_2	d_3	noise	CER (mode 1)	CER (mode 2)	CER (mode3)
40	40	40	3	5	4	4	0(0)	0(0)	0(0)
40	40	40	3	5	4	8	0(0)	0.0095(0.0247)	0.0021(0.0145)
40	40	40	3	5	4	12	0.0038(0.0138)	0.0331(0.0453)	0.0222(0.0520)
40	45	50	3	5	4	4	0(0)	0(0)	0(0)
40	45	50	3	5	4	8	0(0)	0.0042(0.0183)	0(0)
40	45	50	3	5	4	12	0.0062(0.0361)	0.0361(0.0446)	0.0115(0.0422)
40	40	40	4	4	4	4	0(0)	0(0)	0(0)
40	40	40	4	4	4	8	0.0023(0.0165)	0.0034(0.0239)	0(0)
40	40	40	4	4	4	12	0.0519(0.0744)	0.0414(0.0697)	0.0297(0.0644)

Table 1: Given the true d_1, d_2, d_3, the simulation results is calculated across 50 tensors each time.

n_1	n_2	n_3	d_1	d_2	d_3	noise	overall accuracy	estimated d_1	estimated d_2	estimated d_3
40	40	40	3	5	4	4	**1**	3(0)	5(0)	4(0)
40	40	40	3	5	4	8	0.74	3(0)	4.76(0.0610)	3.98(0.02)
40	40	40	3	5	4	12	0.02	2.8(0.0571)	3.58(0.1072)	3.3(0.0915)
40	40	40	3	4	2	4	**1**	3(0)	4(0)	2(0)
40	40	40	3	4	2	8	0.88	3(0)	3.88(0.0464)	2(0)
40	40	40	3	4	2	12	0.6	2.9 (0.0429)	3.56(0.0816)	2 (0)
40	45	50	3	4	2	4	**1**	3(0)	4(0)	2(0)
40	45	50	3	4	2	8	0.94	3(0)	3.94(0.0339)	2(0)
40	45	50	3	4	2	12	0.74	2.96(0.0280)	3.76(0.0732)	2(0)

Table 2: The simulation results across 50 tensors each time from estimating the d_1, d_2, d_3.

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 performace 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. 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 is a extremely low overall accuracy which appears when noise is 12 and the tensoe size is 40*40*40. However, the accuracy is improved quickly as the tensor size is enlarged to 40*45*50. 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 parition 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 3. By looking into each mode separately, as the sample size of that mode increased , the CER of that mode decreased without any exception.

Sparse case. We also test the performace of our approach under different λ when the data is sparse. The $\bar{\lambda}$ in Table 4 is the mean λ we choose across 50 simulations on the same sparsity rate. According to Table 4, 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,

n_1	n_2	n_3	noise	CER(mode 1)	CER(mode 2)	CER(mode3)
40	40	40	4	**0(0)**	**0(0)**	**0(0)**
40	40	40	8	**0(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	**0(0)**	**0(0)**	**0(0)**
40	45	50	8	**0(0)**	0.0027(0.0121)	**0(0)**
40	45	50	12	0.0158(0.0489)	0.0641(0.0629)	0.0336(0.0647)

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

sparsity rate	noise	method	estimated sparsity Rate	Correct Zero Rate	Correct One Rate	Total Correct Rate
0.5	4	$\lambda = 0$	0(0)	0(0)	1(0)	0.5075(0.0676)
0.5	4	$\lambda = 100$	0.5677(0.0667)	1(0)	0.8519(0.0678)	0.9248(0.0377)
0.5	4	$\lambda = 200$	0.5952(0.0688)	1(0)	0.7975(0.0787)	0.8973(0.0433)
0.5	4	$\bar{\lambda} = 86.61396$	0.5606(0.0668)	$\mathbf{0.9993(0.0035)}$	$\mathbf{0.8655(0.0685)}$	$\mathbf{0.9312(0.0377)}$
0.5	8	$\lambda = 0$	0(0)	0(0)	1(0)	0.5075(0.0676)
0.5	8	$\lambda = 100$	0.5072(0.068)	0.879(0.0898)	0.8554(0.0634)	0.8665(0.0559)
0.5	8	$\lambda = 200$	0.5884(0.0618)	0.9753(0.034)	0.7877(0.0776)	0.8794(0.0492)
0.5	8	$\bar{\lambda} = 344.3656$	0.6298(0.0652)	0.9956(0.0128)	0.7259(0.0873)	0.8586(0.0518)
0.8	8	$\lambda = 0$	0(0)	0(0)	1(0)	0.2029(0.0541)
0.8	8	$\lambda = 100$	0.6458(0.0646)	0.7453(0.0616)	0.7136(0.2017)	0.7435(0.0668)
0.8	8	$\lambda = 200$	0.7947(0.0627)	0.9119(0.0601)	0.6259(0.2376)	0.8589(0.0698)
0.8	8	$\bar{\lambda} = 246.9212$	0.826(0.0622)	0.9462(0.0412)	0.6077(0.2495)	0.8841(0.0602)

Table 4: Results for Simulation 4 over 50 simulated data sets ($n_1 = 40, n_2 = 40, n_3 = 40, d_1 = 3, d_2 = 5, d_3 = 4$.

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.

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. **Remember that you can use more than eight pages as long as the additional pages contain *only* cited references.**

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

A Lemma and Theorems

Lemma 1 *Let $\mathbf{Y} \in \mathbb{R}^n$ be a response vector and $\mathbf{X} \in \mathbb{R}^{n \times p}$ the design matrix. Assume the response vector $\mathbf{Y}$ is mean-centered, i.e., $\sum_i Y_i = 0$. Suppose that $\mathbf{X}$ is an orthogonal design matrix with $X^T X = diag(n_1, ..., n_p)$. Define the ordinary least-square estimate $\hat{\boldsymbol{\beta}}_{ols} = (\hat{\beta}_{ols,1}, ..., \hat{\beta}^T_{ols,p})^T$. Consider the following constrained optimization:*

$$\hat{\boldsymbol{\beta}} = argmin\{\frac{1}{2}||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda pen(\boldsymbol{\beta})\}$$

1. Case 1: L-0 penalization. $pen(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_0$:
Under the change of tuning parameter $\lambda' := f(\lambda) = \sqrt{2\lambda}$ such that $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, ..., \hat{\beta}_p)^T$ has a closed-form solution:

$$\hat{\beta}_i = \hat{\beta}_{ols,i}\mathbb{I}_{|\hat{\beta}_{ols,i}| > \frac{\lambda'}{\sqrt{n_i}}} \; for\ all\ i = 1, ..., p$$

2. Case 2: L-1 penalization. $pen(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_1$:
$\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, ..., \hat{\beta}_p)^T$ has a closed-form solution:

$$\hat{\beta}_i = sign(\hat{\beta_{ols,i}})(|\hat{\beta_{ols,i}}| - \frac{\lambda}{n_i})_+ \; for\ all\ i = 1, 2, ..., p$$

Proof 1 *The thing we want to minimize is*

$$L = \frac{1}{2}||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}_0|| = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_0 = L_1 + L_2$$

where $L_1 = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})$, $L_2 = \lambda||\boldsymbol{\beta}||_0$.

Case 1:

Here we view the optimization problem as a case in linear regression. The L_1 is exactly the $RSS/2$ in this case. So we compare the increment of L_1 when L_2 takes different values. We denote z as the number of non-zero elements in $\boldsymbol{\beta}$.
(1) Consider the case we have no constraint on z. Thus we only have to minimize L_1. By the knowledge of linear regression, we know the unique minimizer is $\hat{\boldsymbol{\beta}}_{\boldsymbol{ols}} = (\mathbf{X^T X})^{-1}\mathbf{XY}$. Assume there are m zero elements in $\hat{\boldsymbol{\beta}}_{\boldsymbol{ols}}$ where $0 \leq m \leq p$
(2) Consider the case we have constraint on z: $z = i$, where $i = 0, 1, 2, ..., m$. Obviously, among these cases the L can be minimized if and only if $i = m$. So, $z = m$ and $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_{\boldsymbol{ols}}$ is the minimizer of L when $0 \leq z \leq m$.
(3) Consider the case that we have constraint on x: $z = m + 1$. Then we have to take one more non-zero element in $\boldsymbol{\beta}$ to be zero. Suppose we take $\hat{\beta}_l \neq 0$ to be 0. Then we obtain

$$2L_1 - SSE(\beta_1, ..., \beta_{l-1}, \beta_{l+1}, ..., \beta_p) = SSR(\beta_l)$$

by the columns in $\mathbf{X}$ are orthogonal to each other. Additionally,

$$SSR(\beta_l) = \mathbf{Y}^T(\mathbf{H} - \mathbf{H_l})\mathbf{Y}$$

where $\mathbf{H} = \mathbf{X}(\mathbf{X^T X})^{-1}\mathbf{X} = \sum_{i=1}^{p} \frac{1}{n_i}\mathbf{x_{(i)}}\mathbf{x_{(i)}^T}$, $\mathbf{H_l} = \sum_{i \neq j} \mathbf{x_{(i)}}\mathbf{x_{(i)}^T}$, $\hat{\beta}_l = \frac{1}{n_l}\mathbf{x_l Y}$. Thus, we can simplify the second equation as:

$$SSR(\beta_l) = n_l \hat{\beta}_l^2$$

Thus, by taking $\hat{\beta}_l$ as 0, there is $\frac{n_l \hat{\beta}_l^2}{2}$ increment on L_1, λ decrement on L_2. Obviously, if the increment of L_1 is larger than the decrement L_2, we should not take $\hat{\beta}_l$ as 0; conversely, if the increment of L_1 is less than the decrement of L_2, taking $\hat{\beta}_l$ as 0 can lessen the L.
(4) As we discussed, if there is still at least one element in $\boldsymbol{\beta_k}$ that satisfies that $\frac{n_k \hat{\beta}_k^2}{2} \leq \lambda$, we can keep reducing L by taking $\boldsymbol{\beta_k}$ as 0 until all remain non-zero elements in $\hat{\beta}$ do not satisfy $\frac{n_k \hat{\beta}_k^2}{2} \leq \lambda$. Then we can minimize L.

Over all, the $\boldsymbol{\beta}$ that minimized L is:

$$\hat{\beta}_i = \hat{\beta}_{ols,i} \mathbb{I}_{|\hat{\beta}_{ols,i}| > \frac{\lambda'}{\sqrt{n_i}}} \; for\ all\ i = 1, ..., p$$

Case 2:

Here we use the properties of subderivative. Taking subderivative of L, we obtain

$$\frac{\partial L}{\partial \beta_j} = \begin{cases} \{n_j\beta_j - \mathbf{x_{(j)}^T Y} + \lambda\} & \text{if } \beta_j > 0 \\ [n_j\beta_j - \mathbf{x_{(j)}^T Y} - \lambda, n_j\beta_j - \mathbf{x_{(j)}^T Y} + \lambda] & \text{if } \beta_j = 0 \\ \{n_j\beta_j - \mathbf{x_{(j)}^T Y} - \lambda\} & \text{if } \beta_j < 0 \end{cases}$$

Because β_j minimize L if and only if $0 \in \frac{\partial L}{\partial \beta_j}$ and $\mathbf{X}$ is orthogonal, we get:

$$\hat{\beta}_j = \begin{cases} \frac{\mathbf{x_{(j)}^T Y} + \lambda}{n_j} & \text{if } \hat{\beta}_j < 0 \\ 0 & \text{if } \hat{\beta}_j = 0 \\ \frac{\mathbf{x_{(j)}^T Y} - \lambda}{n_j} & \text{if } \hat{\beta}_j > 0 \end{cases}$$

Here, $\hat{\boldsymbol{\beta}}_{ols} = (\mathbf{X^T X})^{-1}\mathbf{X^T Y} = diag(1/n_1, ..., 1/n_p)X^T Y$, so $\hat{\beta}_{ols,j} = \frac{\mathbf{x_{(j)}^T Y}}{n_j}$. Then the solution of $\hat{\beta}_j$ can be simplified as:

$$\hat{\beta}_i = sign(\hat{\beta_{ols,i}})(|\hat{\beta_{ols,i}}| - \frac{\lambda}{n_i})_+ \; for\ all\ i = 1, 2, ..., p$$

B Algorithm

Algorithm 1 Block Localization

Initialize $c_{11}, c_{12}, ..., c_{1d_1}$, $c_{21}, c_{22}, ..., c_{2d_2}$ and $c_{K1}, c_{K2}, ..., c_{Kd_K}$ by performing one-way k-means clustering on the columns and on the rows of the data matrix X.

repeat

for i in {1,2,...,K} **do**

(a) holding the clusters of all modes fixed, solve $\boldsymbol{\mu}$ by minimizing the loss function with L-0 penalty on $\boldsymbol{\mu}$, that is, $\mu_{r_1,r_2,...,r_K} = S(\frac{\sum_{i=1}^{K}\sum_{l_i \in c_{r_i}} X_{l_1,l_2,...,l_K}}{\prod_{i=1}^{K}|c_{r_i}|}, \frac{\sqrt{2\lambda}}{\sqrt{\prod_{i=1}^{K}|c_{r_i}|}})$

(b) holding $\boldsymbol{\mu}$ and the clusters of other $i-1$ modes fixed, minimizing the loss function with L-0 penalty with respect to $c_{i,1}, ...c_{i,d_i}$ by assigning each observation in mode i to the cluster in mode i whose mean signal is closet to it.

end for

until Convergence