Clustering update issue of the algorithm

Chanwoo Lee

April 5, 2021

1 Algorithm issue

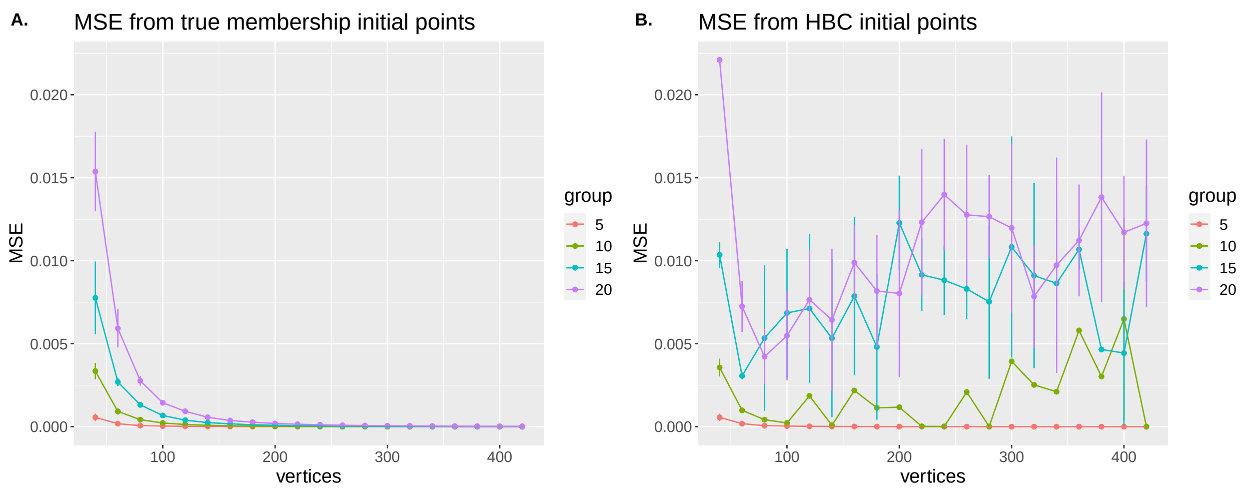

Figure 1: MSE errors depending on the number of vertices and group. Figure A uses ground truth memebership as intial points while B obtains initial points from HSC

Initialization method	Error	Objective	Iteration
HSC	0.0076	571206.5	2 (minimum)
Ground truth	0.00046	547051.1	2 (minimum)
K-means	0.0088	591567.0	2 (minimum)

Table 1: MSE, objective value, and the number of iteration depending on three initialization methods (HSC, ground truth, K-means with nstart = 25).

Currently updated algorithm still have the issue of having poor performance when the number of cluster is high. Figure A shows MSE when ground truth membership is used for initial points while Figure B uses initial points from HSC. Figure 1 implies two possible issues of the algorithm: initialization and update of clustering. From some experiments, I am convinced that the update of clustering in the algorithm does not work well. This explains the unstability of the algorithm when HSC is used because current algorithm does not update cluster membership but remains the same only updating the core tensor. Table 1 shows the error, objective value and the number of iteration according to different initializations. I find out that the number of iteration is always 2 regardless of methods, which implies there is no update in the clustering membership. Indeed, I have checked objective values for each update and it turns out that the clustering membership remains the same. Unless the objective value at initial points is far from the value at the ground truth, the current way of updating core tensor and membership matrices is trapped in local minimum. My guess for this issue is from the fact that the core tensor update $\mathcal{C}^{(t)}$ makes previous membership matrices $\boldsymbol{M}_1^{(t)}, \boldsymbol{M}_2^{(t)}, \boldsymbol{M}_3^{(t)}$ favorable to stay at the next updates for $\boldsymbol{M}_1^{(t+1)}, \boldsymbol{M}_2^{(t+1)}, \boldsymbol{M}_3^{(t+1)}$ as long as the objective value is not too high. I have tried changing the way of updating the membership $\hat{\boldsymbol{M}}_k(a)$ from

$$\hat{\boldsymbol{M}}_k(a) = \arg\min_{r\in[r_k]} \sum_{I_{-k}} \left(\hat{c}_{\hat{\boldsymbol{M}}_1(i_1),\ldots,r,\ldots,\hat{\boldsymbol{M}}_K(i_K)} - y_{i_1,\ldots,a,\ldots,i_K} \right)^2,$$

to

$$\hat{\boldsymbol{M}}_k(a) = \underset{r\in[r_k]}{\arg\min} \left\| \left(\mathcal{M}_k(\mathcal{Y}_k^{(t)})\right)_{j:} - \left(\mathcal{M}_k(\mathcal{S}^{(t)})\right)_{a:} \right\|_2^2 .$$

But this change gives us minor improvement.

2 Sampling probability vs sparsity

Define sampling probability p as

$$\mathbb{P}\left(\mathcal{A}_\omega \text{ is observed }\right) = p,$$

for all $\omega \in E$. Let Ω be an index set whose entries are observed. We decode unobserved entries `NA`. Then, we estimate true signal Θ from incomplete set Ω as

$$\hat{\Theta} = \text{cut}(\tilde{\Theta}), \text{ where } \tilde{\Theta} = \underset{\Theta\in\mathcal{P}_k}{\arg\min} \sum_{\omega\in\Omega} |\mathcal{A}_\omega - p\Theta_\omega|^2.$$

Notice we only evaluate error on observed entries. The major difference from the sparsity parameter is whether the observed tensor is incomplete. If the tensor is complete, this p is sparsity parameter and we evaluate the loss for all entries in E while sampling probability has incomplete tensor and evaluate the loss only for entries in Ω. Therefore, our algorithm needs to adapt incomplete tensor data as an input to follow the *Optimal Estimation and Completion of Matrices with Biclustering Structure.*