Simulation

Jiaxin Hu

To do list:

- Check the Hungarian Algorithm literature.
- Check the iterative post-processing for all matching methods.

1 Simulation Setup and Issues

This simulation aims to evaluate the performances of unseeded matching, seeded matching without non-iterative post-processing, and seeded matching with non-iterative post-processing against the correlation level $\sigma = \sqrt{1-\rho^2}$.

Error. We consider the accuracy measurement

$$err(\hat{\pi}, \pi^*) = \frac{1}{d} \sum_{i \in [d]} \mathbb{1}\{\pi^*(i) \neq \hat{\pi}(i)\}.$$

A smaller err indicates a better matching result.

Remark 1 (Imperfect matching)**.** When $\hat{\pi}$ is not a perfect matching, i.e., we may multiple or no value for $\hat{\pi}(i)$ for some $i \in [d]$. Suppose we have b values for $\hat{\pi}(i)$, $\hat{\pi}(i)_j$ for $j \in [b]$. We define

$$\mathbb{1}\{\pi^*(i) \neq \hat{\pi}(i)\} = \sum_{j \in [b]} \mathbb{1}\{\pi^*(i) \neq \hat{\pi}(i)_j\}.$$

Suppose we have no value for $\hat{\pi}(i)$. Then, let $\mathbb{1}\{\pi^*(i) \neq \hat{\pi}(i)\} = 0$. With imperfect matching $\hat{\pi}$, the err counts the number of wrong pairs that do not occur in π^*.

For example, let $d = 3$ and $\pi^*, \hat{\pi}$ are written as 2-column matrices with rows recording the pairs. Suppose we have true and estimated permutations

$$\pi^* = \begin{bmatrix} 1 & 3 \\ 2 & 2 \\ 3 & 1 \end{bmatrix}, \quad \text{and} \quad \hat{\pi} = \begin{bmatrix} 1 & 3 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}.$$

Here $\hat{\pi}(1) = 3$ or 1, and we have no value for $\hat{\pi}(3)$. Then, we have $err(\hat{\pi}, \pi^*) = \frac{1}{3} \times 2$.

Note that π^* **must** be a perfect matching. Our error satisfies $err(\hat{\pi}, \pi^*) = 1 - acc(\hat{\pi})$, where $acc(\hat{\pi})$ is the accuracy defined in (136) in Ding et al. (2021).

Data generator. We consider the order-3 case. We first generate super-symmetric $\mathcal{A}, \mathcal{B}' \in \mathbb{R}^{d\times d\times d}$ where for all $1 \le i_1 \le i_2 \le i_3 \le d$

$$(\mathcal{A}_{i_1,i_2,i_3}, \mathcal{B}'_{i_1,i_2,i_3}) \sim \mathcal{N}\left((0,0), \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right),$$

and $\mathcal{A}_{i_1,i_2,i_3} = \mathcal{A}_{per(i_1,i_2,i_3)}, \mathcal{B}'_{i_1,i_2,i_3} = \mathcal{B}'_{per(i_1,i_2,i_3)}$ with function $per(\cdot)$ permutes the elements in the vector. Then, we randomly generate π^* and $\mathcal{B} \circ \pi^*$. The outputs of the generator are $\mathcal{A}, \mathcal{B}$ and π^* as a 2-column matrix whose rows recording the matched pairs.

Unseeded matching. Exactly implementing the Algorithm 1. Particularly, we consider the distance d_p with $p = 1$ and $p = 2$, where

$$d_1(i,k) = \frac{1}{d^2}\sum_{j=1}^{d^2} |\mathrm{vec}(\mathcal{A}_{i::})_{(j)} - \mathrm{vec}(\mathcal{B}_{k::})_{(j)}|,$$

$\mathrm{vec}(\mathcal{A}_{i::})_{(j)}$ is the j-th biggest elements in the i-th slice of $\mathcal{A}$, and

$$d_2(i,k) = \frac{1}{d^2}\sqrt{\sum_{t\in\mathcal{A}_{i::}\cup\mathcal{B}_{k::}} \Big(\sum_{i_2,i_3\in[d]^2} \mathbb{1}\{\mathcal{A}_{i,i_2,i_3} \le t\} - \sum_{i_2,i_3\in[d]^2} \mathbb{1}\{\mathcal{B}_{k,i_2,i_3} \le t\}\Big)^2}.$$

Taking input $\mathcal{A}, \mathcal{B}, p$, the algorithm outputs the estimated $\hat{\pi}$ as a 2-column matrix. Note that the output $\hat{\pi}$ may be an imperfect matching without practical meaning.

Seeded matching. Implementing the Algorithm 2 and Sub-Algorithm 1. Here are a few things need to be noticed.

Remark 2 (Non-iterative clean up)**.** Note that there is a non-iterative clean up (not the iterative post-processing after the whole Algorithm 2) inside the Sub-Algorithm 1. The clean up parts are marked as purple in the note 0306. I set a logical input `clean_up = TRUE/FALSE` to indicate whether the algorithm implements the clean up or not.

Remark 3 (Hungarian Algorithm)**.** In Sub-Algorithm 1, we need to solve

$$\tilde{\pi}_1 = \arg\max_{\pi: S^c \mapsto T^c} \sum_{i\in S^c} H_{i,\pi(i)}.$$

In the code, I use the Hungarian Algorithm with function `HungarianSolve()` in package `RcppHungarian`. According to the function document, the function `HungarianSolve()` exactly aims to solve above optimization problem with a fast speed. Though, we need to check the specific reference for the Hungarian Algorithm.

Remark 4 (Seed generation)**.** The most challenging part of the seeded algorithm is the seed generation. Two main problems in the this step

1. Imperfect seeds. Similar with the output in the unseeded algorithm, we may have imperfect seeds, i.e., the seeds is not an one-to-one mapping. Without a perfect seed, we can not implement Sub-Algorithm 1 properly. Therefore, I make a cleaning process for the imperfect

seeds. The cleaning process is pretty tedious (function `clean_seed()`) whose main idea is to choose the "repeated" pairs with smallest d_p. For example, we want to decide which one of the two rows $(1, 2)$, $(1, 3)$ should be kept in the seed. We then check $d_p(1, 2)$ and $d_p(1, 3)$, and keep the one with smaller distance. After the cleaning, we will have a perfect seed. However, the effect of cleaning to the accuracy is unknown.

2. Parameter tuning. Though Theorem 1 indicates that we need to choose $\xi = \mathcal{O}(\sqrt{\log^{1/2} d})$ and $\zeta = \mathcal{O}(\sqrt{\sigma/d^2})$, the choice of constants in front of the d, σ terms effects the seed quality a lot. With improper ξ, ζ, we may have no seeds in the set $\mathcal{S}$ or have too many wrong pairs in $\mathcal{S}$ even after cleaning. Thus, the accuracy for the Algorithm 2 will be seriously effected. For example, the error can drop from 0.9 to 0.3 with $\xi = \sqrt{\log^{1/2} d}$ to $\xi = 1.5 \times \sqrt{\log^{1/2} d}$ for some fixed ζ. Therefore, a grid search for the proper constants may be adapted in future simulations. Currently, I choose $\xi = 5 \times \sqrt{\log^{1/2} d}$ and $\zeta = 3\sqrt{\sigma/d^2}$ which avoids the no seed cases for most settings.

2 Small scale results

Consider three methods: (1) unseeded matching; (2) seeded matching without non-iterative clean up; and (3) seeded matching with non-iterative clean up. We vary the $\sigma \in \{0, 0.1, 0.25, 0.4\}$ which corresponds to $\rho \in \{1, 0.995, 0.968, 0.917\}$ and consider the dimension $d \in \{50, 80\}$. For experiment, we replicate for 5 times, and we report the mean values. See Figure 1.

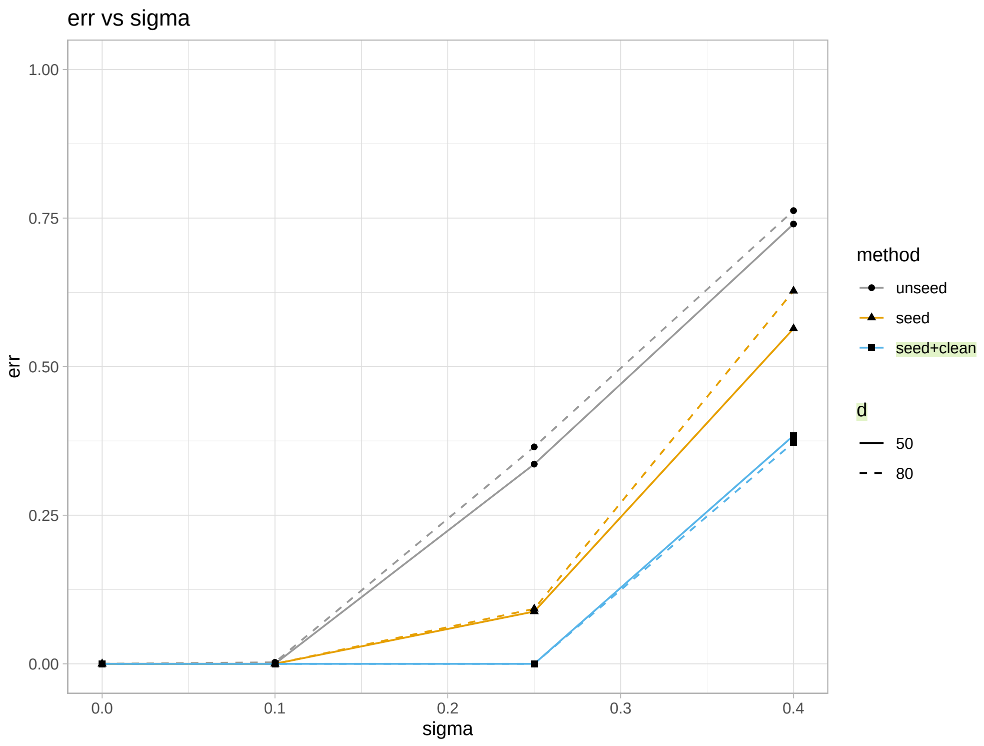

Figure 1: Error versus σ for three matching problem.

In general, the simulation results satisfy the theoretical results. The seeded algorithm improves the matching accuracy, especially with non-iterative clean up. Notice that the errors for unseeded and seeded matching slightly increase with larger dimension. This phenomenon agrees with the signal condition in the theorems that $\sigma \lesssim \frac{1}{\log d}$ and $\sigma \lesssim \frac{1}{\log^{1/2} d}$ for unseeded and seeded matching, respectively. The higher dimensional cases require a stronger signal condition.

References

Ding, J., Ma, Z., Wu, Y., and Xu, J. (2021). Efficient random graph matching via degree profiles. *Probability Theory and Related Fields*, 179(1):29–115.