Research Notes

Xiaoyang Song

Summer 2021

Content

The multi-dimensional graded model

1. Motivation: we want to model the probability that the ith respondent chooses the category $k \in [0, C_j - 1]$ for item j using the information of the latent variables called *factors*, where C_j denotes the number of categories for item j and those categories are indexed by integer from 0 to $C_j - 1$ for simplicity.

2. Notation and model: Suppose that we have N independent respondents, n items and p factors, then define the following:

   (a) i: denotes the respondent i, $i \in [1, N]$.
   (b) j: denotes the item j, $j \in [1, n]$
   (c) C_j: the number of categories of item j.
   (d) p: number of factors
   (e) $\boldsymbol{\beta}_j$: a $p \times 1$ vector of factor coefficient for item j.
   (f) $\mathbf{X}$: a $N \times p$ matrix of factor scores of all respondents.
   (g) $\mathbf{x}_i$: a $p \times 1$ vector of factor scores for respondent i.
   (h) $\mathbf{Y}$: a $N \times n$ matrix of responses of all respondents.
   (i) $\mathbf{y}_i$: the response of respondent i, which is a $n \times 1$ vector.
   (j) y_{ij}: the response of respondent i on item j.
   (k) $\boldsymbol{\alpha}_j$: a $(C_j - 1) \times 1$ vector of intercept term for item j.
   (l) $\boldsymbol{\theta}_j$:=$(\boldsymbol{\beta}_j^\top, \boldsymbol{\alpha}_j^\top)^\top$ vector of all parameters for item j. Suppose $\boldsymbol{\theta}_j$ has dimension $d \times 1$ for simplicity.

   Define the following probabilities: (for $k \in [1, C_j - 1]$)

$$\mathbf{P}(y_{ij} \geq 0 \mid \boldsymbol{\theta}_j, \mathbf{x}_i) = 1 \tag{1}$$
$$\mathbf{P}(y_{ij} \geq C_j \mid \boldsymbol{\theta}_j, \mathbf{x}_i) = 0 \tag{2}$$
$$\mathbf{P}(y_{ij} \geq k \mid \boldsymbol{\theta}_j, \mathbf{x}_i) = \frac{1}{1 + \exp\left(-\boldsymbol{\beta}_j^\top \mathbf{x}_i - \alpha_{jk}\right)} \tag{3}$$

3. Therefore, with above probabilities, we can define the quantity π_{ijk}, which denotes the probability that the ith respondent choose category k for item j:

$$\pi_{ijk} = \mathbf{P}(y_{ij} = k|\boldsymbol{\theta}_j, \mathbf{x}_i) = \mathbf{P}(y_{ij} \geq k|\boldsymbol{\theta}_j, \mathbf{x}_i) - \mathbf{P}(y_{ij} \geq k+1|\boldsymbol{\theta}_j, \mathbf{x}_i) \tag{4}$$

4. With the cell probability π_{ijk} defined, we can write down the following density functions:

   (a) Conditional density function for y_{ij}:

$$f(y_{ij}|\boldsymbol{\theta}_j, \mathbf{x}_i) = \prod_{k=0}^{C_j-1} \pi_{ijk}^{I(y_{ij}=k)} \tag{5}$$

   where $I(y_{ij=k})$ is the indicator function that returns value 1 when $y_{ij} = k$ holds.

   (b) Conditional density function for $\mathbf{y}_i$: $f_1(\mathbf{y}_i|\boldsymbol{\theta}, \mathbf{x}_i) = \prod_{j=1}^{n} f(y_{ij}|\boldsymbol{\theta}_j, \mathbf{x}_i)$.

(c) Now we want to have $\mathbf{y}_i$ only depended on $\boldsymbol{\theta}$ in order to obtain the likelihood function that we want to maximize.If a person's factor scores follows standard multivariate normal distribution , we compute the marginal density by doing integration over $\mathbf{x}_i$:

$$f_2(\mathbf{y}_i|\boldsymbol{\theta}) = \int \prod_{j=1}^{n} f(y_{ij}|\boldsymbol{\theta}_j, \mathbf{x}_i)\Phi(d\mathbf{x}_i) \tag{6}$$

Therefore, the likelihood function of $\boldsymbol{\theta}$ given $\mathbf{Y}$ can be obtained:

$$L(\boldsymbol{\theta}|\mathbf{Y}) = \prod_{i=1}^{N} f_2(\mathbf{y}_i|\boldsymbol{\theta}) = \int \prod_{j=1}^{n} f(y_{ij}|\boldsymbol{\theta}_j, \mathbf{X})\Phi(d\mathbf{X}) \tag{7}$$

(d) If we treat $\mathbf{x}$ as missing data, then it can be rewritten as the following:

$$L(\boldsymbol{\theta}|\mathbf{Z}) = \prod_{i=1}^{N} \left[\phi(\mathbf{x}_i) \prod_{j=1}^{n} f(y_{ij}|\boldsymbol{\theta}_j, \mathbf{x}_i)\right] \tag{8}$$

where $\mathbf{Z} = (\mathbf{Y}, \mathbf{X})$. This is regarded as the complete-data likelihood.

(e) For simplicity, we use $l(\boldsymbol{\theta}|\mathbf{Y})$ and $l(\boldsymbol{\theta}|\mathbf{Z})$ to denote the log-likelihood of the above two likelihood function, respectively.

Difficulties with maximizing the log-likelihood

1. In the multidimensional graded model, we want to maximize $l(\boldsymbol{\theta}|\mathbf{Y})$, where $\boldsymbol{\theta}$ is a $d \times 1$ vector that collects all the parameters that we are going to estimate and $\mathbf{Y}$ is a $n \times N$ matrix of all respondents' response.

2. However, this log-likelihood contains the factor scores $\mathbf{X}$, which is regarded as the missing data. This missing part makes our computation of $\widehat{\boldsymbol{\theta}_{MLE}}$ difficult.

3. Therefore, instead of maximizing the log-likelihood directly, we can make use of the *Fisher's Identity* to do the approximation. There are two algorithms that are mentioned in this paper.

The `EM` algorithm

1. `EM` algorithm stands for the Expectation-Maximization algorithm. This is an iterative algorithm. Let $\boldsymbol{\theta}^{(k)}$ be the value of $\boldsymbol{\theta}$ after the kth iteration. Generally, at the $(k+1)$th iteration, it does the following:

(a) Using $\boldsymbol{\theta}^{(k)}$ and $\mathbf{Y}$ to generate the distribution of the missing data $\mathbf{X}$. We denote this distribution by $\Pi(\mathbf{X}|\boldsymbol{\theta}^{(k)}, \mathbf{Y})$.

(b) Then we compute the conditional expectation of the complete-data log-likelihood over the distribution of $\mathbf{X}$.

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)}) = \int_{\varepsilon} l(\boldsymbol{\theta}|\mathbf{Z})\Pi(d\mathbf{X}|\boldsymbol{\theta}^{(k)}, \mathbf{Y}) \tag{9}$$

where ε denotes a sample space of $\mathbf{X}$.

(c) Then, we maximize $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})$ and obtain $\boldsymbol{\theta}^{(k+1)}$.

We repeat above procedure until $\boldsymbol{\theta}$ converges.

2. Basically, the `EM` algorithm first uses the observed data and the estimated parameter in previous iteration to generate the missing data $\mathbf{X}$ for us, which then enables us to compute the expectation of complete-data log-likelihood and do optimization.

The `MH-RM` algorithm

1. The `MH-RM` algorithm strongly relies on *Fisher's Identity*, which relates the gradient of the log-likelihood with the conditional expectation of the complete-data log-likelihood over $\Pi(\mathbf{X}|\boldsymbol{\theta}, \mathbf{Y})$.

$$\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}|\mathbf{Y}) = \int_{\varepsilon} s(\boldsymbol{\theta}|\mathbf{Z})\Pi(d\mathbf{X}|\boldsymbol{\theta}, \mathbf{Y}) \tag{10}$$

where $s(\boldsymbol{\theta}|\mathbf{Z}) = \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}|\mathbf{Z})$ and ε is some sample space for $\mathbf{X}$.

2. Furthermore, it also relies on the idea of `Robbins-Monro` method, which enables us to iteratively find the root of a particular function $g(\boldsymbol{\theta})$:

(a) The `Robbins-Monro` method is similar to algorithm that finds the roots of a function $g(.)$ by successively approximation, in which, at the $(k+1)$th iteration, the approximation is given by the following:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \gamma g(\boldsymbol{\theta}_k) \tag{11}$$

(b) By introducing a noise term, the `Robbins-Monro` algorithm does the following update in the $(k+1)$th iteration:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \gamma_k R_{k+1} = \boldsymbol{\theta}_k + \gamma_k (g(\boldsymbol{\theta}_k) + \xi_{k+1}) \tag{12}$$

Notation:

i. $\boldsymbol{\theta}_k$: the approximated root for function $g(.)$ at iteration k.

   ii. $R_{k+1} = g(\boldsymbol{\theta}_k) + \xi_{k+1}$: the estimation of $g(\boldsymbol{\theta}_k)$.
   iii. ξ_k: the noise term of estimation of $g(\boldsymbol{\theta}_k)$ at iteration k.
   iv. $g(.)$: a function that we are trying to find its root $\boldsymbol{\theta}$.
   v. γ_k: the gain constant of iteration k.

3. In the `MH-RM` algorithm, both *Fisher's Identity* and `Robbins-Monro` methods are applied.

   (a) Our goal is to maximize the observed-data log-likelihood $l(\boldsymbol{\theta}|\mathbf{Y})$; note that this is equivalent to find the root of $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}|\mathbf{Y})$. Therefore, the hidden logic is to treat the gradient as function $g(.)$ that we mentioned in the `Robbins-Monro` algorithm and find its root iteratively.

   (b) However, since the gradient is hard to evaluate using only the observed data $\mathbf{Y}$, we make use of the *Fisher's Identity*: instead of evaluating the gradient directly, we compute the expected complete-data gradient over $\Pi(\mathbf{X}|\boldsymbol{\theta}, \mathbf{Y})$.

4. Furthermore, the `MH-RM` algorithm uses the curvature information to help converge faster. The complete data information (which is a $d \times d$ matrix) is defined as the following:

$$H(\boldsymbol{\theta} \mid \mathbf{Z}) = -\frac{\partial^2 l(\boldsymbol{\theta} \mid \mathbf{Z})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \tag{13}$$

   which is simply the negative second order derivative of the log-likelihood function.

5. The algorithm starts with $(\boldsymbol{\theta}^{(0)}, \boldsymbol{\Gamma}_0)$. It does the following at the $(k+1)$th iteration:

   (a) Draw m_k missing data $\mathbf{X}_j^{(k+1)}, j \in [1, m_k]$ from a Markov's transition kernel to form a set of complete data $\{\mathbf{Z}_j^{(k+1)} = (\mathbf{X}_j^{(k+1)}, \mathbf{Y}_j) : j \in [1, m_k]\}$.

   (b) Use the sample mean of the complete-data gradient of the log-likelihood to approximate the quantity $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}^{(k)}|\mathbf{Y})$.

$$\mathbf{s}^{(k+1)} = \frac{1}{m_k} \sum_{j=1}^{m_k} \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}^{(k)}|\mathbf{Z}_j^{(k+1)}) \tag{14}$$

   where $\mathbf{s}^{(k+1)}$ denotes the approximated $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}^{(k)}|\mathbf{Y})$.

   (c) Using `Robbins-Monro's` method to update the approximated conditional expectation of the complete-data information matrix:

$$\boldsymbol{\Gamma}_{k+1} = \boldsymbol{\Gamma}_k + \gamma_k \Big\{ \frac{1}{m_k} \sum_{j=1}^{m_k} H(\boldsymbol{\theta}^{(k)} \mid \mathbf{Z}_j^{(k+1)}) - \boldsymbol{\Gamma}_k \Big\} \tag{15}$$

   (d) Update the parameter $\boldsymbol{\theta}^{(k+1)}$ based on the following:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \gamma_k (\boldsymbol{\Gamma}_{k+1}^{-1} \mathbf{s}^{(k+1)}) \tag{16}$$

Notation:

(a) $\mathbf{X}_j^{(k+1)}$: denotes the jth missing data we draw at the $(k+1)$th iteration.

(b) $\mathbf{Y}$: the observed data, which is an $n \times N$ matrix (i.e. the response pattern)

(c) $\mathbf{Y}_j$: the response for the jth person.

(d) $\boldsymbol{\theta}^{(k)}$: the estimated $\boldsymbol{\theta}$ at iteration k.

(e) $\boldsymbol{\Gamma}_k$: the estimated conditional expectation of complete-data information.

(f) γ_k: the gain constant at iteration k.

6. Comments:

   (a) This algorithm makes use of the curvature information to help accelerate the convergence.

   (b) $\boldsymbol{\Gamma}_{k+1}^{-1}\mathbf{s}^{(k+1)}$ works as $\mathbf{R}_{k+1}$ in display (12).

   (c) In step (c), basically what the algorithm does is to treat the difference between the approximated information matrix and the information matrix obtained in previous iteration as the function $g(.)$ in display (12). It uses `Robbins-Monro`'s method to approximate the information matrix as well.

   (d) the convergence of this algorithm can be shown using stability argument about system of differential equations.

Simulation study

1. To examine the performance of the `MH-RM` algorithm, we fit a two-dimensional IFA model to the simulated data set using the algorithm and compare the estimated parameters with the ground truth. In each replication, we first generate the response pattern using the IFA model and then use the algorithm to estimate the parameters.

2. The `mirt` package in `R` is used for this simulation study.

3. Generate the response pattern using the `simdata()` function.

   (a) The data generating model is the logistic IFA model mentioned in the second page with $p = 2$ and $n = 10$ three-category items. In this particular study, the sample size is $N = 1000$.

   (b) The ground truth values of parameters are given. By assuming a multivariate distribution of factor scores $\mathbf{X}$ and using the graded IFA model mentioned before, one can generate the response pattern.

(c) In `R`, the `simdata()` function takes in a vector/matrix of true slopes, a vector/matrix of true intercepts, a number representing the sample size and a parameter that specifies the type of items. In this particular case, the item type is "graded". This function will generate a response pattern of given model and parameters correspondingly.

4. Fit the model using the `mirt` function.

   (a) In this simulation study, the configurations of the `MH-RM` algorithm are given by the following:

      i. The gain constant γ_k is defined as $\gamma_k = \frac{1}{k}$, where k is the iteration number in the algorithm.

      ii. The simulation size m_k is set to one for all iterations.

      iii. Initial values: for all items, set 0.25 for the first intercept and -0.25 for the second intercept; for all slopes but the second slope of the first item, set them to 1.0, and set the second slope of the first item to 0.0.

      iv. Convergence criteria: 1×10^{-4}

5. After 100 replications, the Monte Carlo average of estimates obtained by the MH-RM algorithm is given by the following:

Items	Intercept 1	Intercept 2	Slope 1	Slope 2
1	0.73	-0.69	2.24	NA
2	1.13	-0.09	1.98	0.12
3	-0.11	-1.12	2.53	0.15
4	-0.73	-1.39	1.59	0.09
5	0.54	-0.32	1.67	0.08
6	-0.37	-1.24	1.74	1.30
7	-0.03	-0.93	1.74	1.22
8	1.22	-0.02	1.84	1.30
9	0.20	-0.66	1.50	2.20
10	-1.07	-1.53	1.62	1.62

We can also compute the bias of the estimated values. The bias is defined as the Monte Carlo average of estimates minus the true values.

Items	Intercept 1	Intercept 2	Slope 1	Slope 2
1	0.06	0.03	0.04	NA
2	0.06	0.05	-0.02	0.12
3	0.07	0.10	-0.07	0.11
4	0.03	0.03	-0.01	0.09
5	0.04	0.04	-0.03	0.08
6	0.04	0.02	-0.06	0.10
7	0.04	0.03	-0.06	0.12
8	0.07	0.07	-0.06	0.10
9	-0.07	0.04	-0.10	0.10
10	0.03	0.03	-0.08	0.12

6. Comments

   (a) By observation, the overall trend of my experimental result is that we overestimate slope 2 and underestimates other parameters for all items.

   (b) By observation, the MH-RM algorithm is guaranteed to halt within 2000 cycles for this particular simulation study. Thus, the algorithm gives us converged solution at every Monte Carlo replication. We can also manipulate the `NCYCLE` argument in the `mirt` function to control the maximum number of cycles.

   (c) The `mirt()` function treats the second slope of the last item (item 10) as a fixed value by default. This causes the large bias of the estimates of item 10. This problem can be fixed by adding an argument `pars = "values"` in the `mirt()` function. Then we can modify the object returned by the `mirt()` function.

   (d) The bias we obtained is different from what the paper got. In general, the paper's result is much closer to the ground truth.

   (e) Potential reasons for this discrepancy may be the difference of other implementation details, which are not mentioned in the paper.

7. Resources:

   (a) https://cran.r-project.org/web/packages/mirt/mirt.pdf

   (b) http://www2.uaem.mx/r-mirror/web/packages/mirt/vignettes/mirt-presentation-2012.pdf

Implementing REBOOT on the MIRT model

1. In this particular simulation study, we do the following on each Monte Carlo replication: (To be consistent with the simulation study in Cai's paper, we set $N = 1000$)

   (a) Distribute the whole sample of size N to several, say K, clients each has $n = N/K$ observations.
   (b) On each client, fit a graded IRT model and record the estimated parameters.
   (c) For each client, use the estimated parameter to generate a sample of size nR, where R is a variable that specifies how many bootstrap samples we want to generate for each client.
   (d) Aggregate all the generated samples to one single sample of size NR.
   (e) Use this single sample to fit a graded IRT model and report the estimated parameters.

2. We use two clients for this experiment.The bias of the estimates are given by the following. The bias is defined as the Monte Carlo average of estimates minus the true values.

Items	Intercept 1	Intercept 2	Slope 1	Slope 2
1	0.02	-0.07	0.10	NA
2	0.01	0.00	-0.06	0.07
3	-0.01	-0.02	0.01	0.05
4	0.00	0.00	0.05	0.03
5	0.01	0.00	0.00	0.10
6	0.04	0.03	0.07	0.09
7	0.01	-0.01	-0.08	0.14
8	0.05	0.01	-0.08	0.10
9	0.01	0.02	-0.09	0.11
10	0.02	0.02	-0.04	0.09

3. Comments and observations

   (a) The experiment shows that the REBOOT algorithm works for the graded IRT model. By comparison, we found that the bias that REBOOT obtains are generally higher than those when estimation are done in centralized manner. When increasing the bootstrap sample size, the result become more accurate.
   (b) For identifiability problem, according to the paper, fixing only one slope to zero makes the estimation identifiable for sure.
   (c) By experiment, `mirt()` is a deterministic algorithm. Running for several times on the same dataset gives us the same result.
   (d) The loadings are not orthogonal to each other by experimental results.

4. Understanding the identification problem with examples of $\mathbf{F}$, $\mathbf{L}$, $\mathbf{F}^{'}$, $\mathbf{L}^{'} \in \mathbb{R}^{n\times k}$ such that $\mathbf{LF} = \mathbf{L}^{'}\mathbf{F}^{'}$, $\mathbf{F}^{\top}\mathbf{F} = \mathbf{F}^{'\top}\mathbf{F}^{'} = \mathbf{I}$ but $\mathbf{F} \neq \mathbf{F}^{'}$ and $\mathbf{L} \neq \mathbf{L}^{'}$.

   (a) Suppose that $\mathbf{F}$ and $\mathbf{L}$ be an arbitrary orthogonal matrix of dimension $n \times k$. Then let $\mathbf{F}^{'} = -\mathbf{F}$ and $\mathbf{L}^{'} = -\mathbf{L}$. Then we can check that the above conditions also hold. $\mathbf{L}^{'}\mathbf{F}^{'} = (-\mathbf{L})(-\mathbf{F}) = \mathbf{LF}$ and $\mathbf{F}^{'\top}\mathbf{F}^{'} = (-\mathbf{F})^{\top}(-\mathbf{F}) = \mathbf{I}$. However, by our design, $\mathbf{F} \neq \mathbf{F}^{'}$ and $\mathbf{L} \neq \mathbf{L}^{'}$.

(b) We can also have a direct numerical example. Let $\mathbf{F} = \mathbf{L}^{'} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ and $\mathbf{F}^{'} = \mathbf{L} = \mathbf{I}_{\{2\times 2\}}$. Then we can easily check that $\mathbf{LF} = \mathbf{L}^{'}\mathbf{F}^{'}$, $\mathbf{F}^{\top}\mathbf{F} = \mathbf{F}^{'\top}\mathbf{F}^{'} = \mathbf{I}$.

Understanding the identification problem

1. The use of rotation in factor analysis

   (a) `Varimax` rotation aims to maximize the sum of variance of squared loadings of each factor. In other word, suppose we have a loading matrix $\mathbf{L}$ of dimension $n \times p$, where n is the number of items and p is the number of latent variables, we want to find a rotation matrix $\mathbf{R}$ of dimension $p \times p$ such that $\mathbf{LR}$ has the largest sum of variance of squared loadings of each column.

   (b) Mathematically, the objective function we want to maximize is the following:

$$\mathbf{R}^{*} = \underset{\mathbf{R}}{\operatorname{argmax}} \sum_{j=1}^{p} \{ \frac{1}{n} \sum_{i=1}^{n} [(\mathbf{LR})_{ij}^{2} - \frac{1}{n} \sum_{i=1}^{n} (\mathbf{LR})_{ij}^{2}] \} \tag{17}$$

   Notation:

   i. p: number of latent variables.
   ii. n: number of items.
   iii. $\mathbf{R}$: the rotation matrix.
   iv. $\mathbf{L}$: the loading matrix.

   (c) `Varimax` rotation separate the loadings of factors on each item as far as possible. It can be used as an identification rule.

2. Using rotation as an identification rule

   (a) `Varimax` rotation and `oblimin` rotation are both experimented, and it turns out that the one with `oblimin` rotation has slightly better performance (because its constraint is less strict) in terms of bias and standard error.

   (b) Applying the `oblimin` rotation to the ground truth, we obtain the new ground truth of parameters.

Items	Intercept 1	Intercept 2	Slope 1	Slope 2
1	0.67	-0.72	2.17	0.04
2	1.09	-0.14	1.97	0.04
3	-0.18	-1.22	2.57	0.05
4	-0.76	-1.42	1.58	0.03
5	0.50	-0.36	1.68	0.03
6	-0.41	-1.26	0.68	1.65
7	-0.07	-0.96	0.77	1.52
8	1.15	-0.09	0.78	1.65
9	0.13	-0.70	-0.35	2.86
10	-1.10	-1.56	-0.30	2.05

(c) First, we examine the performance of centralized `mirt`. The settings are the same as what Cai did in his paper. For each monte carlo replication, we rotate the estimated loadings using `oblimin`. After 100 replications, the bias and standard deviation are given by the following:

Items	Intercept 1	Intercept 2	Slope 1	Slope 2
1	0.08(0.15)	0.00(0.15)	0.12(0.19)	-0.10(0.11)
2	0.04(0.14)	0.03(0.12)	-0.14(0.17)	0.13(0.13)
3	0.06(0.12)	0.02(0.14)	-0.02(0.14)	0.03(0.11)
4	0.02(0.10)	0.03(0.11)	-0.04(0.16)	0.05(0.16)
5	0.02(0.09)	0.02(0.11)	-0.05(0.13)	0.07(0.12)
6	0.05(0.10)	0.06(0.13)	-0.04(0.11)	0.05(0.22)
7	0.03(0.07)	0.03(0.07)	-0.09(0.12)	0.06(0.12)
8	0.00(0.12)	0.06(0.12)	-0.09(0.14)	0.07(0.19)
9	0.04(0.14)	0.00(0.14)	-0.09(0.06)	0.07(0.21)
10	0.04(0.11)	0.03(0.14)	-0.11(0.13)	0.00(0.17)

For convenience, the bias and standard deviation of centralized `mirt` without using rotation are given by the following chart:

Items	Intercept 1	Intercept 2	Slope 1	Slope 2
1	0.06(0.14)	-0.01(0.11)	0.09(0.21)	0.00(0.11)
2	0.04(0.13)	-0.03(0.13)	0.01(0.21)	0.12(0.13)
3	0.03(0.11)	0.02(0.14)	-0.03(0.16)	0.17(0.14)
4	0.02(0.09)	0.01(0.09)	-0.02(0.13)	0.08(0.09)
5	0.00(0.11)	-0.01(0.08)	-0.05(0.12)	0.08(0.08)
6	0.04(0.11)	0.02(0.10)	-0.04(0.11)	0.15(0.10)
7	0.03(0.09)	0.03(0.08)	-0.09(0.15)	-0.08(0.08)
8	0.00(0.14)	0.02(0.11)	-0.11(0.16)	0.08(0.11)
9	0.07(0.11)	0.06(0.15)	-0.16(0.18)	0.10(0.15)
10	0.01(0.11)	0.03(0.14)	-0.11(0.17)	0.13(0.14)

(d) For `Reboot` algorithm, in each monte carlo replication, we rotate the estimates of each client. Then, we use the rotated parameters to generate bootstrap samples and refit the model on those samples. The estimates of the first client are used as the starting values for the final estimation.

(e) Specifically, for the `Reboot` algorithm, there are two clients and the bootstrap sample size 10000, which is 10 times of the original sample size. After 100 Monte Carlo replications, the bias and the standard deviation are given by the following:

Items	Intercept 1	Intercept 2	Slope 1	Slope 2
1	-0.01(0.03)	-0.09(0.16)	0.11(0.20)	-0.15(0.18)
2	0.05(0.11)	0.01(0.12)	-0.15(0.07)	-0.14(0.07)
3	-0.01(0.15)	-0.10(0.17)	-0.09(0.20)	-0.07(0.13)
4	-0.08(0.12)	-0.12(0.13)	-0.01(0.06)	-0.01(0.05)
5	0.09(0.14)	0.04(0.12)	0.01(0.12)	0.13(0.19)
6	-0.06(0.09)	0.11(0.14)	-0.09(0.12)	0.05(0.10)
7	0.01(0.13)	-0.07(0.11)	0.06(0.06)	0.13(0.20)
8	0.05(0.05)	-0.07(0.08)	-0.01(0.13)	-0.15(0.10)
9	-0.07(0.16)	0.09(0.16)	0.12(0.19)	0.05(0.21)
10	-0.09(0.13)	-0.08(0.18)	0.10(0.22)	0.07(0.29)

We compare the estimates given by the `Reboot` algorithm with the estimates of the first clients. The bias of the estimates given by the first clients is given by the following:

Items	Intercept 1	Intercept 2	Slope 1	Slope 2
1	-0.07(0.11)	-0.10(0.12)	-0.08(0.14)	-0.17(0.16)
2	0.09(0.07)	0.03(0.12)	0.17(0.15)	-0.20(0.11)
3	-0.11(0.14)	-0.09(0.22)	-0.07(0.07)	-0.13(0.14)
4	-0.07(0.12)	-0.14(0.21)	0.10(0.09)	-0.11(0.12)
5	-0.05(0.16)	-0.08(0.23)	0.06(0.11)	0.09(0.21)
6	0.11(0.05)	-0.09(0.08)	-0.03(0.18)	0.11(0.13)
7	0.14(0.14)	0.11(0.11)	0.11(0.09)	0.12(0.15)
8	0.13(0.07)	-0.04(0.07)	-0.09(0.13)	-0.11(0.09)
9	-0.06(0.11)	0.12(0.17)	0.12(0.14)	0.10(0.24)
10	-0.18(0.17)	0.05(0.24)	0.09(0.21)	0.11(0.27)

(f) Observations and comments

i. According to the experimental results, we find that rotation does not improve the overall model performance in terms of bias and variance. This is consistent with theoretical expectation since rotation itself does not change the underlying model in nature. Rotation can be understood as another representation for the model.

ii. In practice, rotations are often used for better interpreting the model.

Experiments on REBOOT and full-sample MIRT

(a) Note that in this simulation study, the basic setup strictly follows the one mentioned in Cai's paper, and factor rotation, which is discussed in the previous section, is not used for all experiments.

(b) As for metrics that are used to measure performance, sinTheta distance is calculated to examine whether an algorithm performs well on estimating the slopes of the underlying model mentioned in previous sections. As for statistical errors of the intercepts, the Frobenius norm of difference of the estimators are calculated.

(c) In order to compare the performance between `Reboot` and centralized `MIRT`, the experiments are designed in two regimes.

   i. The first is that we tried different number of clients on `Reboot` algorithm with the global sample size fixed. Note that when increasing the number of clients, the local sample size per clients decreases. We need to make sure that the local sample is enough for `MIRT` to converge. For this particular simulation study where there are 39 free parameters, a local sample size of above 100 guarantees that it generates reasonable results.

   ii. On the other hand, we tried to increase the number of clients with local sample size fixed, which means the global sample size will increase. For this simulation study, we choose to fix the local sample size to 125.

According to above the design, the results of experiments with different setups are given below: (Note that errors for slopes and intercepts are calculated and reported separately, and all results are rounded to four decimal places)

Experiment setup: fixed global sample size: $N = 1000$

K	2	4	5	8	10
Single client estimator	0.3715	0.5177	0.5542	0.7896	0.8752
Average estimator	0.2651	0.2923	0.3469	0.4221	0.4506
`REBOOT` estimator	0.2762	0.2875	0.3127	0.3176	0.3155
`MIRT` estimator	0.2582				

Table 1: sinTheta distance for estimators of slopes

Experiment setup: fixed global sample size: $N = 1000$

K	2	4	5	8	10
Single client estimator	0.7915	1.1012	1.3416	1.9153	2.1240
Average estimator	0.5475	0.5751	0.6078	0.8510	0.9279
`REBOOT` estimator	0.5599	0.5644	0.5518	0.5610	0.5835
`MIRT` estimator	0.5383				

Table 2: Frobenius norm of difference of intercept estimators

(d) The experimental results of fixing global sample size to $N = 2500$ are given below:

Experiment setup: fixed global sample size: $N = 2500$

K	2	5	10	20	25
Single client estimator	0.2307	0.3630	0.5228	0.7943	0.8942
Average estimator	0.1816	0.1818	0.2141	0.3228	0.3770
`REBOOT` estimator	0.1851	0.1832	0.2028	0.2075	0.2214
`MIRT` estimator	0.1780				

Table 3: sinTheta distance for estimators of slopes

(e) The results of experiments show that all three types of estimators share a trend of increasing error of estimation as the number of clients increase. This is reasonable because the decrease of local sample size increases the difficulties of local estimation. In addition, the average estimator tends to work better `REBOOT` estimator when K is not so big. However, when K increases, the change of error of estimation for `REBOOT` estimator is relatively stable than that of the average estimator, and at certain point, `REBOOT` estimator starts to outperform the average estimator.

Experiment setup: fixed global sample size: $N = 2500$

K	2	5	10	20	25
Single client estimator	0.4710	0.7902	1.1151	2.0704	2.1280
Average estimator	0.3235	0.3398	0.4021	0.6928	0.7339
`REBOOT` estimator	0.3565	0.3508	0.3554	0.3874	0.4124
`MIRT` estimator	0.3233				

Table 4: Frobenius norm of difference of intercept estimators

(f) Experiments with fixed local sample size are conducted with a local sample size of $n = 125$. By observation, $n = 125$ is a sample size for local estimators to provide reasonable results while maintain that the local sample is not huge. Different K values ($K = 2, 4, 8, 12, 16, 20$) are tried to make comparison. Additionally, the performances of `MIRT` estimators with corresponding global sample size are also calculated as a benchmark.

(g) The experimental results are given below:

Experiment setup: fixed local sample size: $n = 125$

K	2	4	8	12	16	20
Average estimator	0.6935	0.5803	0.4221	0.3803	0.3433	0.3228
`REBOOT` estimator	0.5343	0.4229	0.3176	0.2620	0.2381	0.2075
`MIRT` estimator	0.5038	0.3948	0.2582	0.2210	0.2105	0.1780

Table 5: sinTheta distance for estimators of slopes

(h) Some plots of the above experimental results are given below for better demonstration of the pattern:

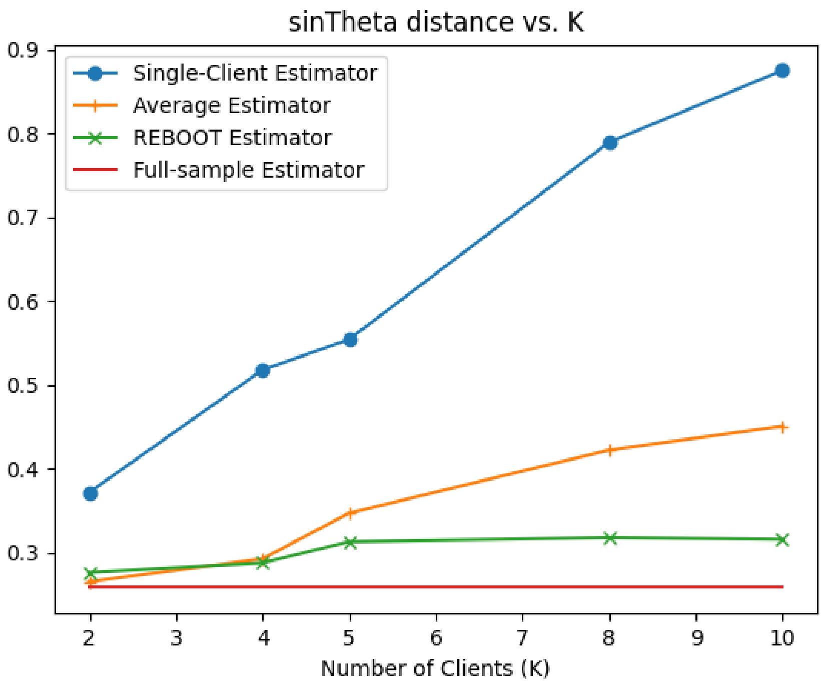

Figure 1: Fixed Global Sample size: $N = 1000$

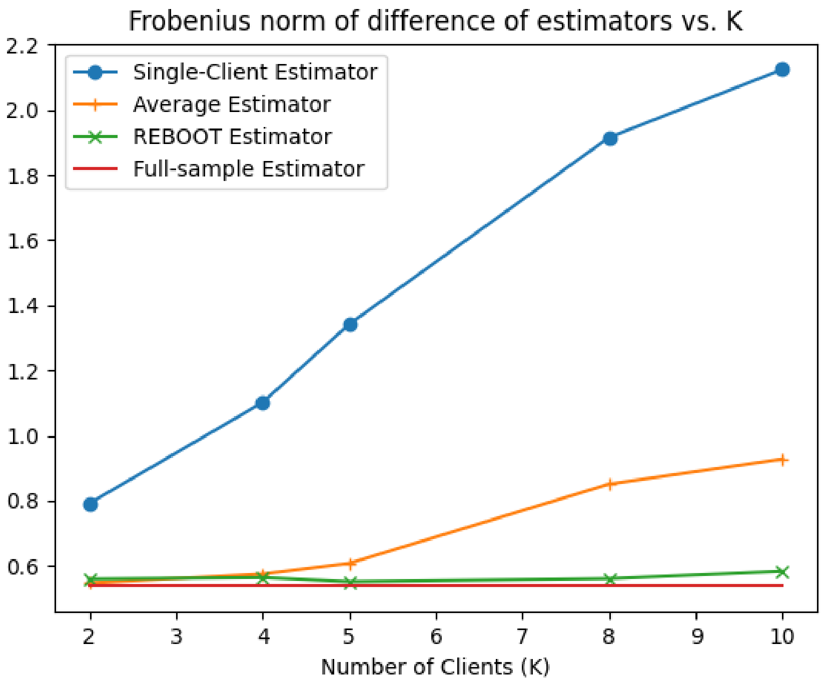

Figure 2: Fixed Global Sample size: $N = 1000$

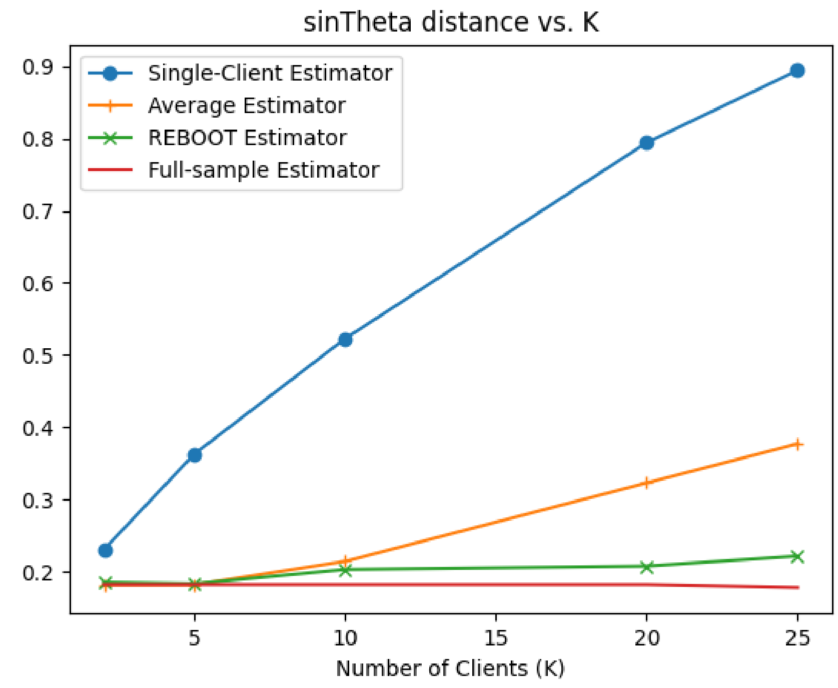

Figure 3: Fixed Global Sample size: $N = 2500$

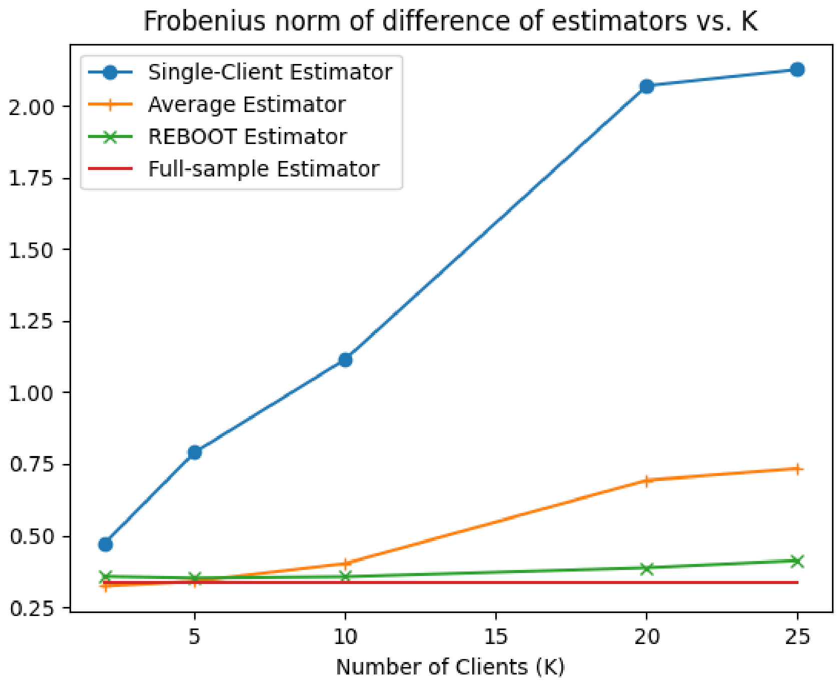

Figure 4: Fixed Global Sample size: $N = 2500$

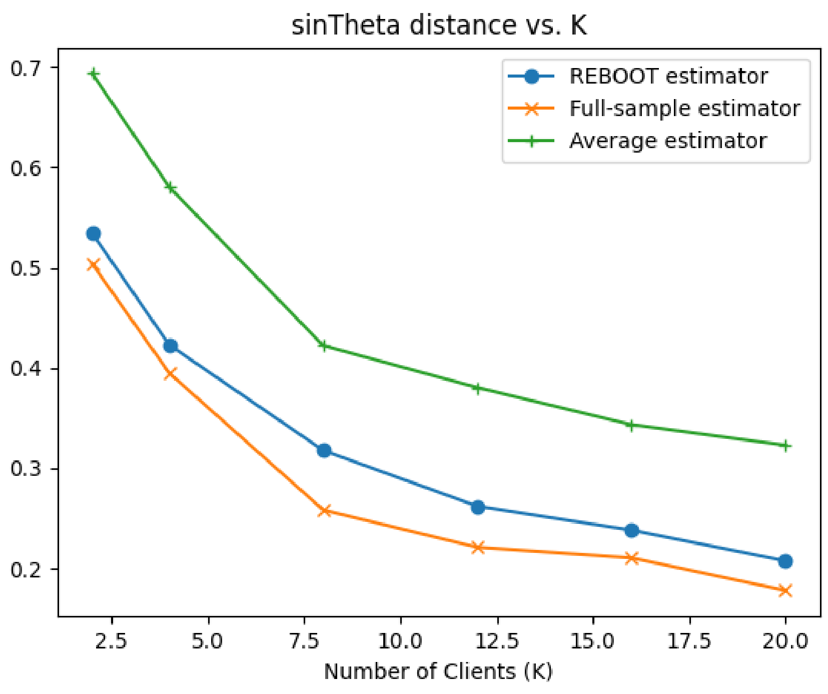

Figure 5: Fixed Local Sample size: $n = 125$

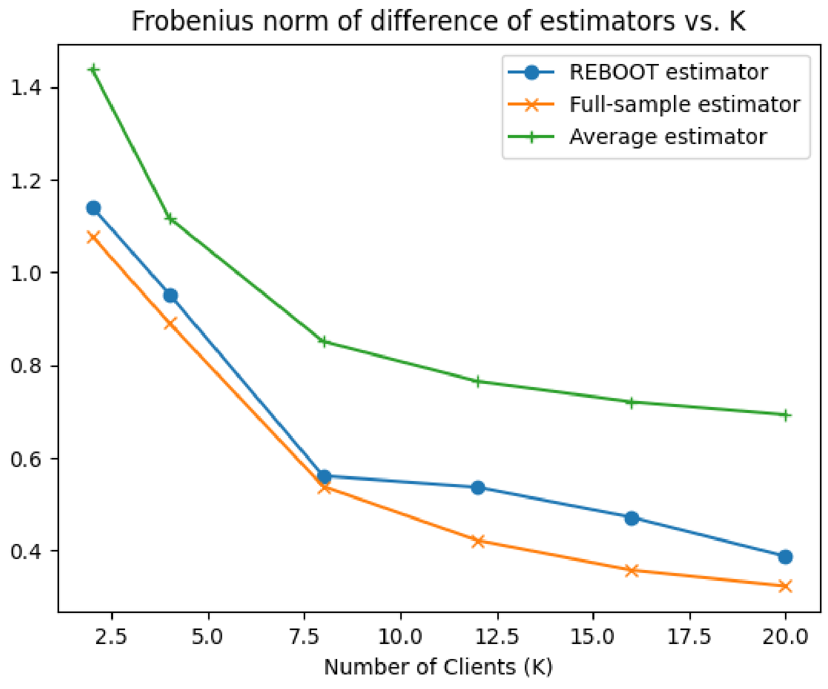

Figure 6: Fixed Local Sample size: $n = 125$

Experiment setup: fixed local sample size: $n = 125$

K	2	4	8	12	16	20
Average estimator	1.4382	1.1175	0.8510	0.7652	0.7208	0.6928
`REBOOT` estimator	1.1412	0.9534	0.5610	0.5365	0.4717	0.3874
`MIRT` estimator	1.0784	0.8910	0.5383	0.4218	0.3575	0.3233

Table 6: Frobenius norm of difference of intercept estimators

(i) Comments:

i. From experimental results, a local sample size of less than 100 could result in really poor estimation for each client; therefore, a local sample size of 125 is chosen, and the second regime of the experiment is conducted based on $n = 125$.

ii. `Reboot` estimator tends to have a better and more stable performance than the average estimator and the single-client estimator. However, consistent to the theoretical expectation, the full-sample estimator works the best.

PLT constraint for loadings in MIRT model

1. The PLT (positive-diagonal lower-triangular) constraint of the loading matrix guarantees the identifiability of the estimation of MIRT model.

   (a) Notations:

      i. m: the number of latent factors

      ii. p: the number of items

      iii. $\mathbf{\Lambda} = \{\lambda_{ij}\}_{i\in[1,p],j\in[1,m]}$: the estimated loading matrix of dimension $p \times m$

   (b) By definition of PLT constraint, imposing the PLT constraint on $\mathbf{\Lambda}$ implies that the first m rows of $\mathbf{\Lambda}$, which is a $m \times m$ square matrix, is a lower triangular matrix with positive diagonal entries.

   (c) For intercepts term in the estimation, note that there does not exist rotational indeterminacy so no identification constraint is needed.

2. The PPLT (Permutation PLT) constraint is the less restricted version and also the common used constraint for loading matrix.

(a) Instead of requiring the first m rows of $\mathbf{\Lambda}$ to satisfy the PLT constraint, the PPLT constraint claims that as long as a $m \times m$ square matrix, which is any arbitrary subset of m rows of $\mathbf{\Lambda}$, satisfies the PLT constraint, then $\mathbf{\Lambda}$ is identifiable.

(b) Let $\mathbf{r} = \{r_i\}_{i \in [1,m]}$ be an index vector, where $r_i \in [1,p]$ and $r_i \neq r_j$ for $i \neq j$. Note that $\mathbf{r}$ indicates which rows are picked to form the square PLT matrix.

(c) Mathematically, if the loading matrix is identifiable, then there exists $\mathbf{r}$ such that $\lambda_{r_i j} > 0$ for $i \in [1,m]$, and $\lambda_{r_i j'} = 0$ for $j' > j$. Basically, this ensures that the submatrix is PLT.

(d) Imposing PPLT is sufficient to guarantee identifiability of the loading matrix. Anderson and Rubin (1956) mentioned in their paper that the model is identifiable if the loading matrix satisfies the General Triangularity condition.

   i. General Triangularity condition: there exists a $p \times m$ matrix $\mathbf{A}$ such that $\mathbf{A}^\top \mathbf{\Lambda} = \mathbf{L}$, where $\mathbf{L}$ is a PLT matrix of dimension $m \times m$.

   ii. In PPLT constraint, $\mathbf{A}$ is just the row permutation matrix, which is defined by $\mathbf{r}$: $A_{ij} = 0$ if $r_j = i$, and $A_{ij} = 1$ otherwise. By doing $\mathbf{A}^\top \mathbf{\Lambda}$, we are just finding the desired submatrix $\mathbf{L}$, which is PLT.

   iii. Therefore, we find that $\mathbf{\Lambda}$ satisfies the General Triangularity condition if we imposed the PPLT constraint on that. And PPLT is sufficient to guarantee identifiability.

3. Tentative thoughts on the second regime of federated learning on MIRT

   (a) Instead of dividing among samples (i.e. test takers), we can also divide on items (i.e. questions). However, there exists several potential problems:

      i. Aggregation by concatenation: since we estimate the loadings separately, we can only concatenate those when aggregating.

      ii. How to generate data:

         A. Generate data using loadings of all items: in this way, estimating only a subset of items' loadings using that will not make sense.

         B. Generate data using loadings of selected items: in this way, we are basically estimating the parameters independently, there is meaningless to do federated learning.

      iii. For identifiability issue, we need to impose the PLT constraint on the loading matrix separately; however, by doing this, it is possible that the latent structure of the full-sample loadings are different from what are estimated in federated manner.

Generalized Latent Factor Model

Model Setting

Suppose that we have N people and J items (i.e. questions), and each question has C possible categories. We define the following notations:

1. $Y_{ij} \in [0, C-1]$: the response of the ith person on item j, $i \in [0, N-1]$, $j \in [0, J-1]$.
2. K: number of latent factors
3. $\boldsymbol{\theta}_i = (\theta_{i1}, ..., \theta_{iK})^\top \in \mathbb{R}^{K\times 1}$: the factor scores for the ith person.
4. $\boldsymbol{\Theta} \in \mathbb{R}^{N\times K}$: the factor score matrix
5. $\mathbf{a}_j = (a_{j1}, ..., a_{jK})^\top \in \mathbb{R}^{K\times 1}$: the loading vector for item j.
6. $\mathbf{A} \in \mathbb{R}^{J\times K}$: the loading matrix

The generalized latent factor model assumes that Y_{ij} given $\boldsymbol{\theta}_i$ and $\mathbf{a}_j$ is a member of the exponential family with density function:

$$f(Y_{ij} = y \mid \mathbf{a}_j, \boldsymbol{\theta}_i) = exp\big(\frac{ym_{ij} - b(m_{ij)}}{\phi} + c(y, \phi)\big)$$

where $b()$. and $c(.)$ are two functions that decide which distribution in the exponential family the model distribution belongs to, ϕ is the scale parameter, and $m_{ij} = \mathbf{a}_j^\top \boldsymbol{\theta}_i$. Correspondingly, the joint likelihood is defined as the following:

$$L(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_N, \mathbf{a}_1, ..., \mathbf{a}_J) = \prod_{i=1}^{N}\prod_{j=1}^{J} f(Y_{ij} = y_{ij} \mid \mathbf{a}_j, \boldsymbol{\theta}_i)$$

Definition. A random variable X is said to be in the exponential family if its density function can be written in the following form:

$$f_X(x \mid \theta) = h(x)\ exp(\eta(\theta)T(x) + A(\theta))$$

where $h(.)$, $\eta(.)$, $T(.)$ and $A(.)$ are some functions.

Definition. Consider a nonempty parameter space $S \subset \mathbb{R}^{\mathbb{Z}^+\times\{1,...K\}} \times \mathbb{R}^{\mathbb{Z}^+\times\{1,...J\}}$ for $(\boldsymbol{\Theta}, \mathbf{A}) \in S$. Then for the kth factor, it is structurally identifiable in S if for any $(\boldsymbol{\Theta}, \mathbf{A})$, $(\widehat{\boldsymbol{\Theta}}, \widehat{\mathbf{A}})$ with $P_{\boldsymbol{\Theta},\mathbf{A}} = P_{\widehat{\boldsymbol{\Theta}},\widehat{\mathbf{A}}}$, $\sin_+\angle(\boldsymbol{\Theta}_{[k]}, \widehat{\boldsymbol{\Theta}}_{[k]}) = 0$

Questions

1. The paper mentions that the identification can be achieved by imposing constraints on the design matrix $\mathbf{Q}$; this works well for the Confirmatory Factor Analysis where we impose the relationship between latent factors by changing $\mathbf{Q}$. However, for explortory factor analysis, for example, the scenario in Cai's paper, how can we achieve identifiability? the paper mentions that rotation is a technique but is that sufficient?

`DistributedPCA` algorithm

1. In the `DistributedPCA` algorithm, we use the average of the projection matrix of top K eigenspaces of all clients to estimate the full-sample covariance matrix. Then, on the central processer, we calculate the top K eigenvectors as final output.

2. The pseudocode is given by the following:

   (a) Compute $\widehat{\mathbf{V}}_K^{(l)}$ of $\widehat{\mathbf{\Sigma}}^{(l)}$ locally for each machine $l \in [1, m]$

   (b) Compute $\widetilde{\mathbf{\Sigma}} = \frac{1}{m}\Sigma_{l=1}^m \widehat{\mathbf{V}}_K^{(l)} \widehat{\mathbf{V}}_K^{(l)\top}$, and output its top K eigenvectors $\widetilde{\mathbf{V}}$.

   (c) If eigenvalues are needed, compute $\widehat{\mathbf{\Lambda}}^{(l)} = \mathrm{diag}(\widetilde{\mathbf{V}}^\top \widehat{\mathbf{\Sigma}}^{(l)} \widetilde{\mathbf{V}})$, and then take the average $\widetilde{\mathbf{\Lambda}} = \frac{1}{m}\Sigma_{l=1}^m \widehat{\mathbf{\Lambda}}^{(l)}$, where $\{\lambda_i\}_{i=1}^K = \{\mathbf{\Lambda}_{ii}\}_{i=1}^K$.

   Notation:

   (a) $\mathbf{x}_i^{(l)} \in \mathbb{R}^{d\times 1}$: a single feature vector of local machine l

   (b) $\mathbf{X}^{(l)} \in \mathbb{R}^{N\times d}$: feature matrix of local machine l

   (c) $\widehat{\mathbf{\Sigma}}^{(l)} = \frac{1}{n}\widehat{\mathbf{X}}^{(l)\top}\widehat{\mathbf{X}}^{(l)} \in \mathbb{R}^{d\times d}$: sample covariance matrix of local machine l

   (d) $\widehat{\mathbf{V}}_K^{(l)} \in \mathbb{R}^{d\times K}$: estimated top K eigenvectors on local machine l

   (e) $\widehat{\mathbf{\Lambda}}^{(l)} \in \mathbb{R}^{K\times K}$: estimated eigenvalues on local machine l

3. Adopting `DistributedPCA` on the Generalized Latent Factor model.

4. Variance-Bias Decomposition

   (a) The statistical error analysis of `DistributedPCA` is conducted by analyzing the variance and the bias term respectively. The statistical error used is the sinTheta distance. For example, the sinTheta distance between $\mathbf{V}_K$ and $\widetilde{\mathbf{V}}_K$ is given by the following:

$$\rho(\mathbf{V}_K, \widetilde{\mathbf{V}}_K) = \|\mathbf{V}_K\mathbf{V}_K^\top - \widetilde{\mathbf{V}}_K\widetilde{\mathbf{V}}_K^\top\|_F$$

(b) By variance and bias decomposition:

$$\rho(\mathbf{V}_K, \widetilde{\mathbf{V}}_K) \leq \rho(\mathbf{V}_K^\star, \widetilde{\mathbf{V}}_K) + \rho(\mathbf{V}_K^\star, \mathbf{V}_K)$$

where $\mathbf{V}_K^\star \in \mathbb{R}^{d\times K}$ is the expectation of the estimated top K eigenvectors. In order to analyze the statistical error for these quantities, we need to make use of the Davis-Kahan sinTheta Theorem, which provides an error bound for the sinTheta distances.

(c) The **Davis-Kahan Theorem**
Theorem. Let Σ, $\widehat{\Sigma} \in \mathbb{R}^{d\times d}$ be two symmetric matrix with eigenvalues $\{\lambda_i\}_{i=1}^d$ and $\{\widehat{\lambda}_i\}_{i=1}^d$ respectively. (Note that these two sets are sorted in descending order) Similarly, let $\mathbf{V}$, $\widehat{\mathbf{V}} \in \mathbb{R}^{d\times K}$ be the corresponding eigenvector matrix; and each column represents an eigenvector. Now, fix $1 \leq r \leq s \leq K$ and let $\mathbf{V}_{[r:s]}$, $\widehat{\mathbf{V}}_{[r:s]} \in \mathbb{R}^{d\times(s-r+1)}$ be the subsets of $\mathbf{V}$, $\widehat{\mathbf{V}}$, respectively. Then define the following:

$$\delta := \inf\{|\ \widehat{\lambda} - \lambda\ | : \lambda \in [\lambda_s, \lambda_r] \text{ and } \widehat{\lambda} \in (-\infty, \widehat{\lambda}_{s-1}] \cup [\widehat{\lambda}_{r+1}, +\infty)\}$$

where $\widehat{\lambda}_0 = -\infty$ and $\widehat{\lambda}_{K+1}$ are defined.
With the above definitions and notation, we have the following error bound for sinTheta distance between $\mathbf{V}_{[r:s]}$ and $\widehat{\mathbf{V}}_{[r:s]}$:

$$\|\rho(\mathbf{V}_{[r:s]}, \widehat{\mathbf{V}}_{[r:s]})\|_F \leq \frac{\|\Sigma - \widehat{\Sigma}\|_F}{\delta}$$

where $\mathbf{V}_{[r:s]} = (\mathbf{v}_r, ..., \mathbf{v}_s)$, and $\mathbf{v}_i$ is the ith column of $\mathbf{V}$.

Aggregation Regime I: QR Decomposition

1. We can impose the identification constraint on the local MIRT estimator before aggregating them to compute the global estimator. The detailed algorithm is following:

   (a) Run MIRT on each client separately.

   (b) For each local estimator, do QR decomposition and take the corresponding $\mathbf{R}$ matrix of QR decomposition as the final local estimator.

   (c) Compute the global estimator by directly taking the average of local estimators.

2. QR Decomposition

   (a) QR Decomposition decomposes a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ into a product $\mathbf{A} = \mathbf{QR}$ of an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{m\times m}$ (i.e. $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$) and an upper triangular matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$.

   (b) QR decomposition can be determined using the Gram-Schmidt process.

Aggregation Regime II: `DistributedPCA` on MIRT

1. Suppose there are N individuals, J items, and K latent factors (with $K < J$), we can adopt the aggregation mechanism in `DistributedPCA` on the factor model by doing the following:

   (a) Compute $\widehat{\mathbf{V}}^{(l)}$ of $\widehat{\mathbf{\Sigma}}^{(l)} = \widehat{\mathbf{A}}^{(l)}\widehat{\mathbf{A}}^{(l)\top}$ locally for each machine $l \in [1, m]$

   (b) Compute $\widetilde{\mathbf{\Sigma}} = \frac{1}{m}\Sigma_{l=1}^{m}\widehat{\mathbf{V}}^{(l)}\widehat{\mathbf{V}}^{(l)\top}$ and its eigenvectors $\widetilde{\mathbf{V}}$

   (c) Compute $\widehat{\mathbf{\Lambda}}^{(l)} = \operatorname{diag}(\widetilde{\mathbf{V}}^{\top}\widehat{\mathbf{\Sigma}}^{(l)}\widetilde{\mathbf{V}})$ and $\widetilde{\mathbf{\Lambda}} = \frac{1}{m}\Sigma_{l=1}^{m}\widehat{\mathbf{\Lambda}}^{(l)}$

   (d) Compute $\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^{\top} = \widetilde{\mathbf{V}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{V}}^{\top}$

   (e) Compute $\widetilde{\mathbf{A}} = \widetilde{\mathbf{V}}\widetilde{\mathbf{\Lambda}}^{\frac{1}{2}}$ (see discussion below)

   Important Notation:

   (a) $\mathbf{A} \in \mathbb{R}^{J\times K}$: the ground truth loading matrix.

   (b) $\widehat{\mathbf{A}}^{(l)} \in \mathbb{R}^{J\times K}$: the local estimator of loading matrix on machine l.

   (c) $\widetilde{\mathbf{A}} \in \mathbb{R}^{J\times K}$: the global estimator of loading matrix.

   (d) $\widehat{\mathbf{V}}^{(l)} \in \mathbb{R}^{J\times J}$: the eigenvectors of $\widehat{\mathbf{\Sigma}}^{(l)}$.

   (e) $\widetilde{\mathbf{V}} \in \mathbb{R}^{J\times J}$: the eigenvectors of $\widetilde{\mathbf{\Sigma}}$.

   (f) $\widehat{\mathbf{\Lambda}}^{(l)} \in \mathbb{R}^{J\times J}$: the recovered eigenvalues of $\widehat{\mathbf{\Sigma}}^{(l)}$.

   (g) $\widetilde{\mathbf{\Lambda}} \in \mathbb{R}^{J\times J}$: the recovered eigenvalues.

2. Factorization of $\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^{\top} \in \mathbb{R}^{J\times J}$

   (a) Cholesky Decomposition

      i. Cholesky decomposition decomposes a real symmetric, positive definite matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ into a lower triangular matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$ and its transpose $\mathbf{R}^{\top} \in \mathbb{R}^{n\times n}$, (i.e. $\mathbf{M} = \mathbf{R}\mathbf{R}^{\top}$).

      ii. However, by experiments, in this simulation study that follows setup in Cai's paper, $\mathbf{M} = \widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^{\top}$ is not necessary to be a positive definite (PD) matrix. (Note that for the ground truth $\mathbf{A} \in \mathbb{R}^{J\times K}$, $\operatorname{rank}(\mathbf{A}\mathbf{A}^{\top}) \leq K$) Furthermore, our target $\mathbf{A}$ is not a square matrix as well. Therefore, Cholesky decomposition is not a suitable factorization approach for this algorithm.

   (b) Instead, $\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^{\top}$ can be decomposed in the following way:

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^{\top} = \widetilde{\mathbf{V}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{V}}^{\top} = \widetilde{\mathbf{V}}\widetilde{\mathbf{\Lambda}}^{\frac{1}{2}}\widetilde{\mathbf{\Lambda}}^{\frac{1}{2}}\widetilde{\mathbf{V}}^{\top} = (\widetilde{\mathbf{V}}\widetilde{\mathbf{\Lambda}}^{\frac{1}{2}})(\widetilde{\mathbf{\Lambda}}^{\frac{1}{2}\top}\widetilde{\mathbf{V}}^{\top}) = (\widetilde{\mathbf{V}}\widetilde{\mathbf{\Lambda}}^{\frac{1}{2}})(\widetilde{\mathbf{V}}\widetilde{\mathbf{\Lambda}}^{\frac{1}{2}})^{\top}$$

where $\widetilde{\mathbf{A}} = \widetilde{\mathbf{V}}\widetilde{\mathbf{\Lambda}}^{\frac{1}{2}} \in \mathbb{R}^{J\times J}$. Note that the rank of $\mathbf{A}$ is less or equal to K so we can manipulate the recovered diagonal matrix of eigenvalues $\widetilde{\mathbf{\Lambda}}$ by only preserving the first K diagonal entries such that it has at most K non-zero eigenvalues. If this is done, then we can only take the first K columns of $\widetilde{\mathbf{A}}$ and enforce $\widetilde{\mathbf{A}} \in \mathbb{R}^{J\times K}$.

However, to enforce the identifiability constraint, we need to do a QR decomposition after obtaining $\widetilde{\mathbf{A}}$. The final global estimator is the corresponding $\mathbf{R}$ matrix of QR decomposition.

3. Experimental Results

Experiment setup: $N = 1000$

# of clients	2	4	5	8	10
QR estimator	0.7567	0.7942	0.8872	1.0321	1.1578
`Distributed-PCA` estimator	1.6453	1.7266	1.7610	1.7418	1.7905

Table 7: sinTheta Distances between estimated loadings and ground truth

(a) The experimental results suggest that the factorization approach for the `DistributedPCA` algorithm might be problematic. By inspection of the estimated loadings, we found that the estimation deviates from the ground truth significantly and the estimation results do not capture any of the internal structure of latent factors.

(b) In this experiment, in order to determine the proper way to impose identification constraint, initial conditions which are mentioned in Cai's paper

Items	Slope 1	Slope 2
1	-0.04	0.00
2	1.22	0.73
3	-1.56	-0.00
4	-2.20	0.71
5	-1.45	-0.31
6	-0.47	-1.62
7	0.92	-1.63
8	-2.06	0.38
9	0.37	-2.10
10	-0.58	-1.50

Table 8: Bias of the estimated loadings from `DistributedPCA`

are removed. Unlike in Cai's experiment setup where we have 39 parameters to estimate, in this case, we have to estimate all 40 parameters. No prior information is provided in this experiment.

(c) Fixed Implementation problems:

   i. The recovered diagonal matrix of eigenvalues (i.e. $\widetilde{\mathbf{\Lambda}}$) is not necessarily sorted in descending order so taking the first two elements as the largest eigenvalues is problematic. This issue is fixed by manually sorting the eigenvalues after recovering.

   ii. The formula to recover the eigenvalues for each local machine is fixed to be $\widehat{\mathbf{\Lambda}}^{(l)} = \mathrm{diag}(\widetilde{\mathbf{V}}^\top \widehat{\mathbf{\Sigma}}^{(l)} \widetilde{\mathbf{V}})$.