STATS 489 Final Report

Weihan Xu

Introduction

Cognitive Diagnosis Models (CDMs) are popular statistical tools widely applied to educational assessments and psychological diagnoses, which has been receiving increasingly more attention in the past two decades. CDMs model the relationship between the test items and the examinees' latent skills, which is helpful in assessment design and post-assessment analysis of the examinees' latent attribute patterns. In this project, we use CDM packages and RBM methods to overcome the computational difficulties in estimating CDMs.[1]

Related Work

1. Mathematics and Statistics Background:

   ✧ Notations:

   $\overline{\boldsymbol{\alpha_\iota}}$: Attribute profile. The entry is zero is the student doesn't have that attribute. Otherwise, the entry is one.

   $\boldsymbol{\theta}$: a $J \times 2^k$ matrix. $\theta_{j,\overline{\alpha_\iota}}$: The probability of answering j-th question correctly if student i has the attribute $\overline{\alpha_\iota}$.

   $\boldsymbol{R}$: The student responses

   Q matrix: a $J \times K$ matrix. J is the total number of items and K denotes the total number of attributes.

   Assumptions:

   $$\Pr(correct|have\ all\ skills\ needed) > \Pr(correct|\ have\ some\ skills\ needed)$$
   $$\geq \Pr\ (correct|don't\ have\ any\ skills\ needed.)$$

   ✧ Models:

   <u>DINA model:</u>

   Assumptions:

[1] Jimmy de la Torre(2009), "DINA Model and Parameter Estimation", Journal of Educational and behavioral statistics, https://journals.sagepub.com/doi/abs/10.3102/1076998607309474

1) Students having one skill may not necessarily to learn another skill faster. That is, there is no interaction effect due to α_k and $\alpha_{k\prime}$.
2) Students have to get all the skills that are needed to answer the question correctly. This is what $\delta_{j_{1,2,3....k_j^*}}\, \pi_{k=1}^{K_j^*}\alpha_{lk}$ denotes.
3) DINA model considers the probability of guessing and slipping and $Pr(slipping) \ll Pr(guessing)$. That is, $\theta_{j,\bar{\alpha}} = ((1-s_j)^{I(\bar{\alpha}\geq\theta_{j,*})})\cdot (g_j)^{1-I(\bar{\alpha}\geq\theta_{j,*})}$
4) DINA model has two latent group. Within one group, the probability of success is the same.[2]

Mathematical Formula:

$$P_j(\overline{\alpha_\iota}) = \delta_{j_0} + \delta_{j_{1,2,3....K_j^*}}\, \pi_{k=1}^{K_j^*}\alpha_{lk}$$

where

δ_{j0} is the intercept for item j. It indicates the probability of a correct response when none of the required attributes is present. It is typically non-negative;

$\delta_{j_{1,2,3....K}}$ indicates the intersection effect due to α_1 , …, $\alpha_{K_j^*}$ and K_j^* indicates the number of attributes needed.

<u>GDINA model: (Generalized DINA model)</u>

Assumptions:

1) Examinees with fewer required attributes for an item can have a higher probability of answering the item correctly.
2) GDINA model has $2^{K_{j^*}}$ latent groups where $K_{j^*} = \sum_{i=0}^{K} q_{jk}$

Mathematical Formula:

$$Pr\,(\overline{\boldsymbol{\alpha_{\iota J}}}) = \delta_{j_0} + \sum_{k=1}^{K_j^*}\delta_{jk}\alpha_{lk} + \sum_{k\prime=k+1}^{K_j^*}\sum_{k=1}^{K_j^*-1}\delta_{jkk\prime}\,\alpha_{lk}\alpha_{lk\prime} + \cdots + \delta_{j_{1,2,3....k_j^*}}\, \pi_{k=1}^{K_j^*}\alpha_{lk}$$

where

δ_{j0} is the intercept for item j. It indicates the probability of a correct response when none of the required attributes is present. It is typically non-negative;

[2] Xu and Shang, "Identifying Latent Structures in Restricted Latent Class Models", https://par.nsf.gov/servlets/purl/10049844

δ_{jk} is the main effect due to α_k. It is typically non-negative;

$\delta_{jkk\prime}$ is the interaction effect due to α_k and $\alpha_{kk\prime}$. It can take any values.

$\delta_{j_{1,2,3\ldots K}}$ indicates the intersection effect due to α_1 , …, $\alpha_{K_j^*}$ and K_j^* indicates the number of attributes needed.

Method:

1. Programming techniques:

R and python are two main programming language involved in this project. CDM packages developed by Alexander Robizsch, Thomas Kiefer, Ann Cathrice George and Ali Uenlue are used for both data. The codes written by Chengcheng(RBM code) are used here to make comparisons.

2. Process

a) Data simulation:

True Q-matrix:

To ensure the Q-matrix is identifiable so that it can be learned from the observational data, we specify it as follows:

$$Q = \begin{bmatrix} I_K \\ Q_1 \\ Q_2 \end{bmatrix},$$

Where I_k is a K dimensional identity matrix; $Q_1 \in \{0,1\}^{K\times K}$ with value 1 in the (i,i)th entries for I = 1,…,K and the (i, i+1)th entries for i=1, …, K-1, and values 0 for all the other entries. $Q_2 \in \{0,1\}^{K\times K}$ with value 1 in the entries (i, i) for i = 1, …, K, (i, i-1) for i = 2, …K and (i, i+1) for i, …, K-1, and value 0 for all the remaining entries. This construction sets the number of items to be J=3K.

Both independent and dependent settings of latent attributes are explored. We use two steps to simulate the latent patterns. First, a Gaussian latent vector is generated for each subject $z_i = (z_{i1}, \ldots, z_{iK}) \sim N(0, \sum)$(iid) for i = 1,…., N, where

$$\Sigma = (1-\rho)\mathbf{1}_K + \rho\mathbf{1}_K\mathbf{1}_K^\top,\ \mathbf{1}_K = (1,\ldots,1)_K^\top,$$

and ρ is the correlation between any two different latent attributes. In particular, for a given K, we specify the thresholds ranging from -0.5 to 0.5, with a step size of 1/(K-1), for each attribute 1,2, … K respectively. Then $\alpha_{ik} = 1$ if z_{iK} is greater than its respective threshold and $\alpha_{ik} = 0$ otherwise. We will also consider two uncertainty levels, guessing parameter and slipping parameter in the model.

b) Computation:

- CDM packages:

We will initialize Q matrix to be all ones at the very beginning and use returned deltas to recover Q matrix. The Q recovery code is attached to this report at the end. The details of the implementation: I will use the delta values that returned by CDM::gdina package to recover the Q matrix. We first check the delta value of the interaction terms and then the delta value of the terms that represent one attribute. If the delta of the interaction term is larger than zero, then those two attributes will all have one in Q matrix. Else, the entry in Q-matrix will be set to zero.

Then we will use the CDM tools: order Q, to reorder the columns in this matrix. This process is using Hungarian algorithm. The main goal of this algorithm is to jointly minimize the total column-wise matching errors.

Cross Validation and repetition (parallel computing) are used to get a more general result. [3]

- Restricted Boltzmann Machines:

RBM are generative models that can learn probabilistic distributions over a collection of inputs. We denote the visible units by students' responses and hidden units by attributes. The key assumption is that we only consider hidden-visible connection only. The model parameters, denoting by $\theta = \{b, c, W\}$, are visible units bias, hidden units bias and weights indicating the intersection between two kinds of nodes. Our goal is to minimize:

$$\min_{\boldsymbol{\theta}} - \log\left\{P(\boldsymbol{R};\boldsymbol{\theta})\right\} + \lambda \sum_{j=1}^{J}\sum_{k=1}^{K} |w_{j,k}|.$$

where $\log\{P(R;\theta)\}$is the probability of true response given θ, λ is a tuning parameter for the L_1 penalty. Contrastive Divergence algorithm is used to find the optimal values. When doing the

[3] Xu and Shang, "Identifying Latent Structures in Restricted Latent Class Models", https://par.nsf.gov/servlets/purl/10049844

cross validation, we are looking for the Q matrix that gives us the lowest mean batch error. The mean batch error is the reconstruction error between the latest sampled visible batches and the original observed batches. [4]

c) Evaluation: (Please see the code on Appendix)

- Entry-wise overall percentage error (OE):

$$\mathrm{OE} := \frac{1}{JK}\sum_{j=1}^{J}\sum_{k=1}^{K}\mathbb{1}\{\hat{q}_{j,k} \neq q_{j,k}\},$$

- True positives percentage error (OTP):

$$\mathrm{OTP} := \frac{\sum_{j=1}^{J}\sum_{k=1}^{K}\mathbb{1}\{\hat{q}_{j,k} = 0, q_{j,k} = 1\}}{\sum_{j=1}^{J}\sum_{k=1}^{K}\mathbb{1}\{q_{j,k} = 1\}},$$

- True negatives percentage error (OTN):

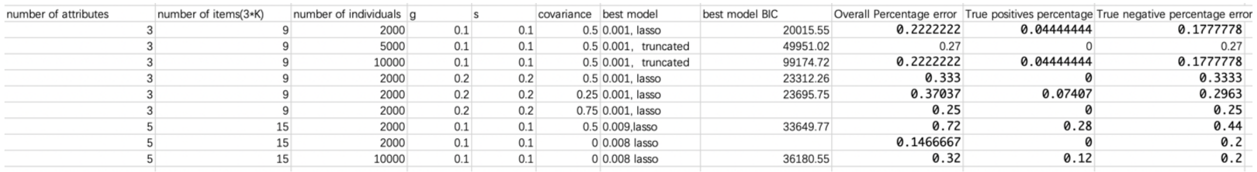
$$\mathrm{OTN} := \frac{\sum_{j=1}^{J}\sum_{k=1}^{K}\mathbb{1}\{\hat{q}_{j,k} = 1, q_{j,k} = 0\}}{\sum_{j=1}^{J}\sum_{k=1}^{K}\mathbb{1}\{q_{j,k} = 0\}},$$

Experiments and results:

The experiments that I did through this semester: local Experiments and Large experiments with the help of Greatlakes.

1. <u>Local Experiments:</u> I firstly did some local experiments to get a rough idea of the method and helping us to get some hypothesis about the trend.

1) I firstly compute Q matrix with GDINA package locally with a small number of repetitions. Three repetitions are made here. The following table is the result that I have:



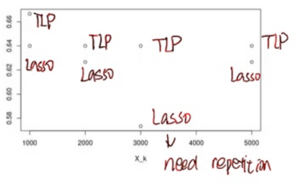
number of attributes	number of items(3∗K)	number of individuals	g	s	covariance	best model	best model BIC	Overall Percentage error	True positives percentage	True negative percentage error
3	9	2000	0.1	0.1	0.5	0.001, lasso	20015.55	0.2222222	0.04444444	0.1777778
3	9	5000	0.1	0.1	0.5	0.001, truncated	49951.02	0.27	0	0.27
3	9	10000	0.1	0.1	0.5	0.001, truncated	99174.72	0.2222222	0.04444444	0.1777778
3	9	2000	0.2	0.2	0.5	0.001, lasso	23312.26	0.333	0	0.3333
3	9	2000	0.2	0.2	0.25	0.001, lasso	23695.75	0.37037	0.07407	0.2963
3	9	2000	0.2	0.2	0.75	0.001, lasso		0.25	0	0.25
5	15	2000	0.1	0.1	0.5	0.009,lasso	33649.77	0.72	0.28	0.44
5	15	2000	0.1	0.1	0	0.008 lasso		0.1466667	0	0.2
5	15	10000	0.1	0.1	0	0.008 lasso	36180.55	0.32	0.12	0.2

(The column "best model" is got after comparing the best result returned by "lasso" regularization and "TLP" regularization. The criterion that I used to choose model is BIC.)

When other variables keep constant, when we increase covariance, I find a quite unusual trend is that the overall percentage error drops when K=3 but it is the opposite when K=5. When other variables keep constant, when we increase g and s, we can see that the overall error increases.

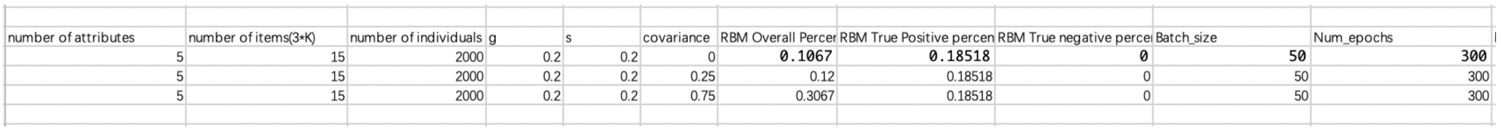
[4] Li, Ma and Xu(2021), "Learning Large Q-matrix by Restricted Boltzmann Machines", https://arxiv.org/pdf/2006.15424.pdf.

~~~
```{r, include=FALSE}
X_k <-c(1000,1000,2000,2000,3000,3000,5000,5000)
plot(y=OE_list_N,x=X_k)
```
~~~



When other variables keep constant, when we increase the number of respondents, the overall error will slightly decrease. This is quite intuitive because when we increase the N, we have more information. Therefore, we are more likely to get a good result. We can also find that TLP will give us a better result.

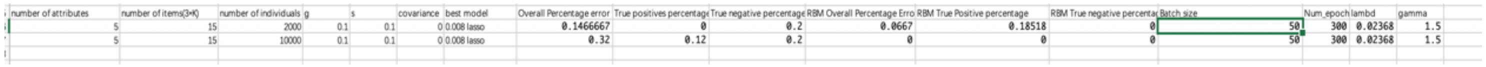
2) I also compute Q matrix with RBM method locally with a small number of repetitions to get a rough idea of the method and the trend.



number of attributes	number of items(3*K)	number of individuals	g	s	covariance	RBM Overall Percer	RBM True Positive percen	RBM True negative perce	Batch_size	Num_epochs
5	15	2000	0.2	0.2	0	0.1067	0.18518	0	50	300
5	15	2000	0.2	0.2	0.25	0.12	0.18518	0	50	300
5	15	2000	0.2	0.2	0.75	0.3067	0.18518	0	50	300

When other variables keep constant, we can see when we increase the covariance value, we can see that the overall percentage error increases. This is quite intuitive, though different from the trend that we get in CDM package method.

3) I also want to get a rough idea of how these two methods perform under the same settings.



number of attributes	number of items(3*K)	number of individuals	g	s	covariance	best model	Overall Percentage error	True positives percentage	True negative percentage	RBM Overall Percentage Erro	RBM True Positive percentage	RBM True negative percenta	Batch size	Num_epoch	lambd	gamma
5	15	2000	0.1	0.1	0	0.008 lasso	0.1466667	0	0.2	0.0667	0.18518	0	50	300	0.02368	1.5
5	15	10000	0.1	0.1	0	0.008 lasso	0.32	0.12	0.2	0	0	0	50	300	0.02368	1.5

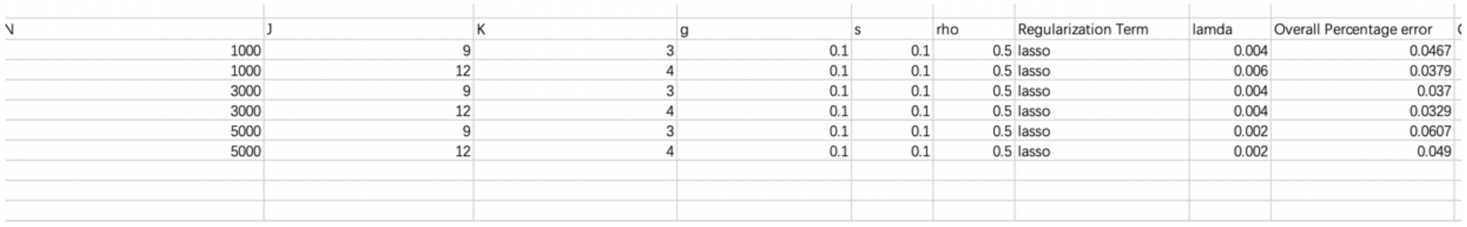
Though RBM method is used to approximate CDM method, we can find the RBM method gives us a better result when other variables keep constant.

2. <u>Doing large experiments with the help of Greatlakes.</u>

After getting some hypothesis about the trend, I start to run the experiments for more repetition to get a more general result. This is done with the help of Greatlakes.

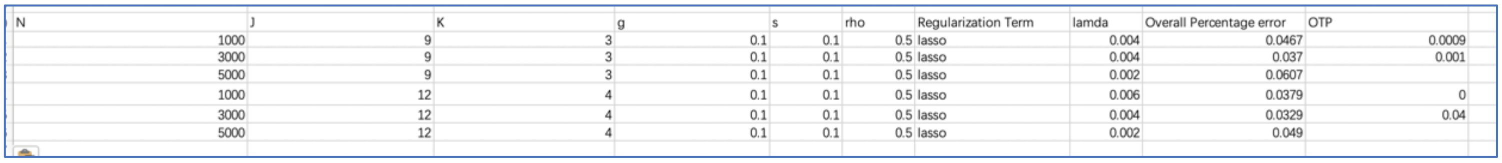
<u>CDM:: Lasso:</u>

I grid search the best parameters within the range $\lambda \in \{0, 0.002, 0.004, 0.006, 0{,}008, 0.01\}$ and choose the model with the lowest BIC. I will choose the regularization term to be lasso first.



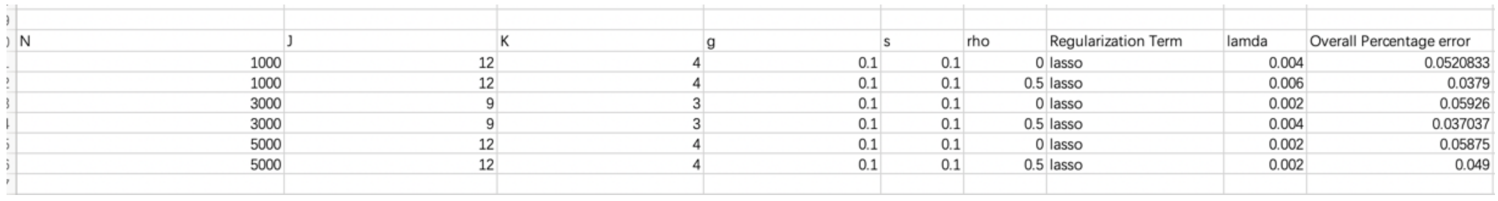
N	J	K	g	s	rho	Regularization Term	lamda	Overall Percentage error
1000	9	3	0.1	0.1	0.5	lasso	0.004	0.0467
1000	12	4	0.1	0.1	0.5	lasso	0.006	0.0379
3000	9	3	0.1	0.1	0.5	lasso	0.004	0.037
3000	12	4	0.1	0.1	0.5	lasso	0.004	0.0329
5000	9	3	0.1	0.1	0.5	lasso	0.002	0.0607
5000	12	4	0.1	0.1	0.5	lasso	0.002	0.049

- ➢ When I keep other variables constant, the higher the K is, the lower the overall error is.



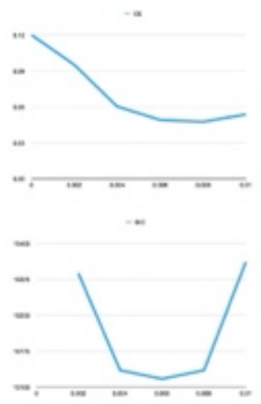
N	J	K	g	s	rho	Regularization Term	lamda	Overall Percentage error	OTP
1000	9	3	0.1	0.1	0.5	lasso	0.004	0.0467	0.0009
3000	9	3	0.1	0.1	0.5	lasso	0.004	0.037	0.001
5000	9	3	0.1	0.1	0.5	lasso	0.002	0.0607	
1000	12	4	0.1	0.1	0.5	lasso	0.006	0.0379	0
3000	12	4	0.1	0.1	0.5	lasso	0.004	0.0329	0.04
5000	12	4	0.1	0.1	0.5	lasso	0.002	0.049	

- ➢ When I keep other variables constant and increases N, the overall percentage error first increases then decreases. We can also see that the value of λ that gives us the best model decreases. When we decrease λ, we will have more restriction on the estimate values and might result in more bias. As a result, the overall percentage might go up.

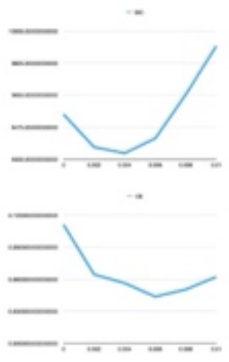


N	J	K	g	s	rho	Regularization Term	lamda	Overall Percentage error
1000	12	4	0.1	0.1	0	lasso	0.004	0.0520833
1000	12	4	0.1	0.1	0.5	lasso	0.006	0.0379
3000	9	3	0.1	0.1	0	lasso	0.002	0.05926
3000	9	3	0.1	0.1	0.5	lasso	0.004	0.037037
5000	12	4	0.1	0.1	0	lasso	0.002	0.05875
5000	12	4	0.1	0.1	0.5	lasso	0.002	0.049

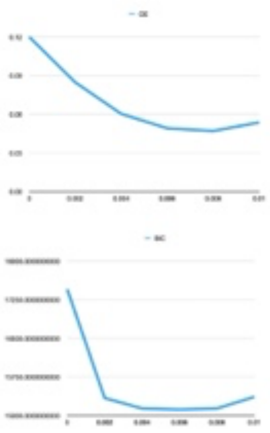
- ➢ When I keep other variables constant, the higher the correlation between attributes is, the lower the overall percentage error is.
- ➢ We can also see that overall true negative percentage error is higher than the overall true positive percentage error.
- ➢ How lambda influences BIC and overall percentage error.
  From all the plots below, we can see that they all have the same trend:
  As lambda increases, the overall percentage error firstly decreases and then increases. As lambda increases, BIC firstly decreases and then increases.



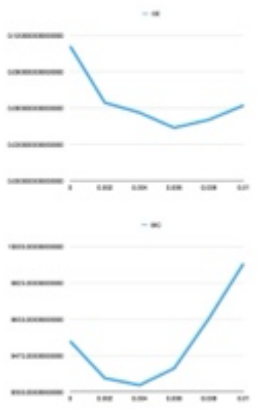
N=1000, J=15, K=5, Rho=0.5, lasso, g=s=0.1.



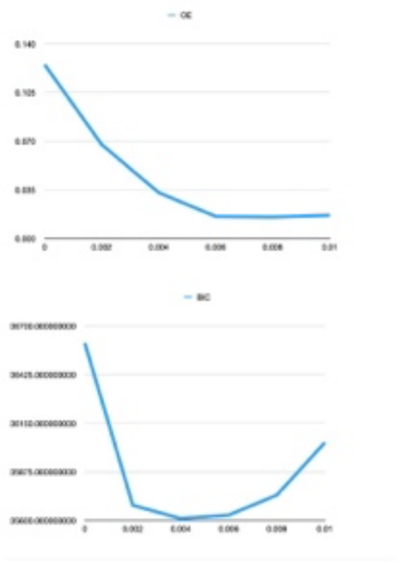
N=1000, J=9, K=3, Rho=0, lasso, g=s=0.1



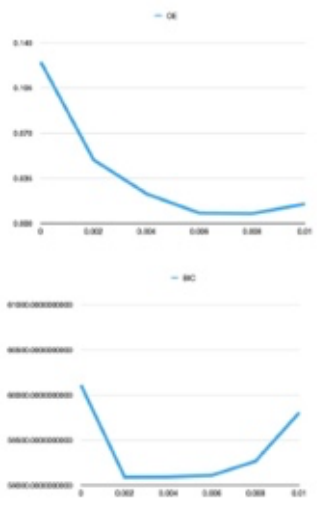
N=1000, J=15, K=5, Rho=0.5, lasso, g=s=0.1.



N=1000, J=9, K=3, Rho=0, lasso, g=s=0.1



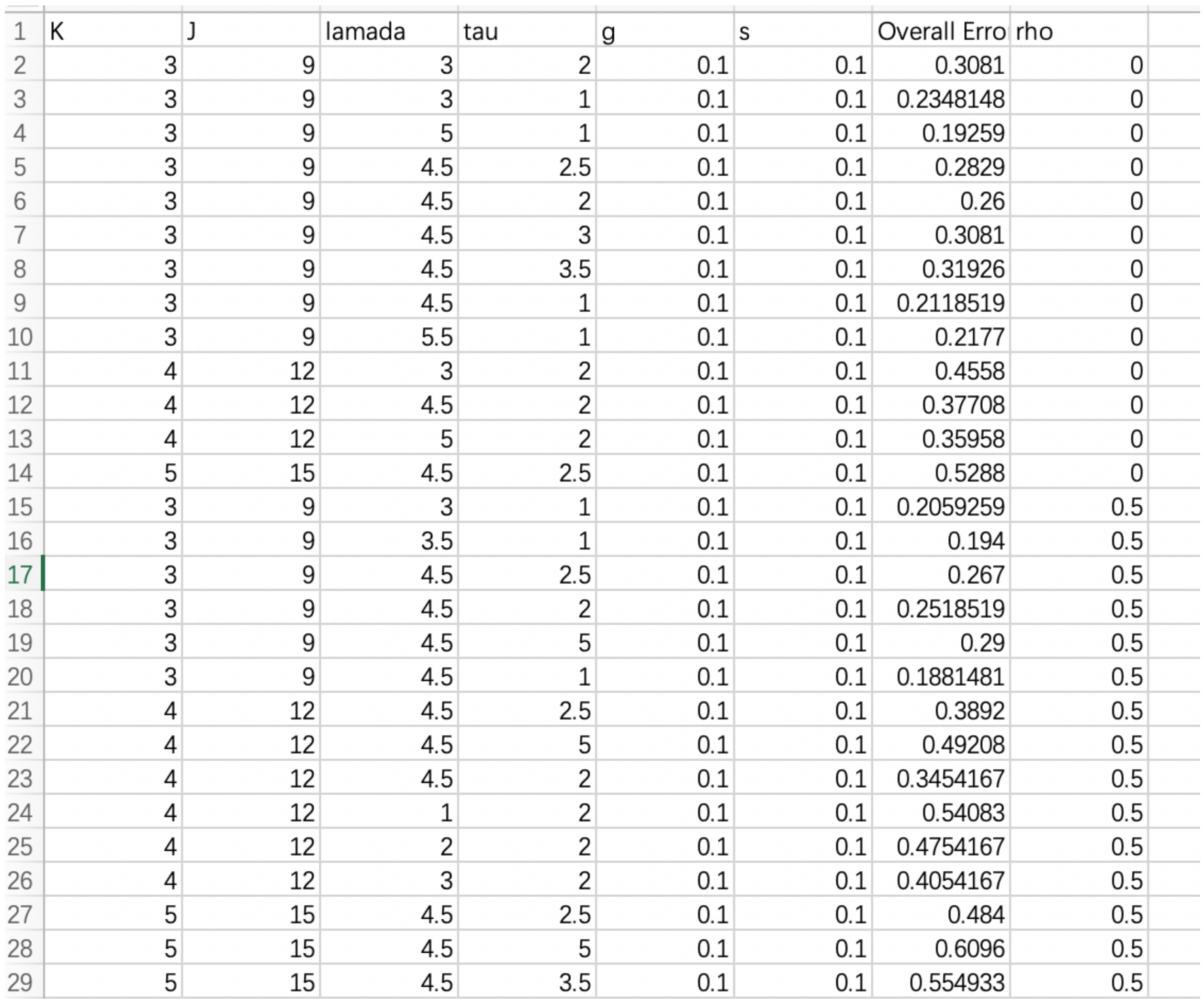
N=3000, J=12, K=4, Rho=0.5, lasso, g=s=0.1



N=5000, J=12, K=4, Rho=0.5, lasso, g=s=0.1

CDM:: TLP regularization:

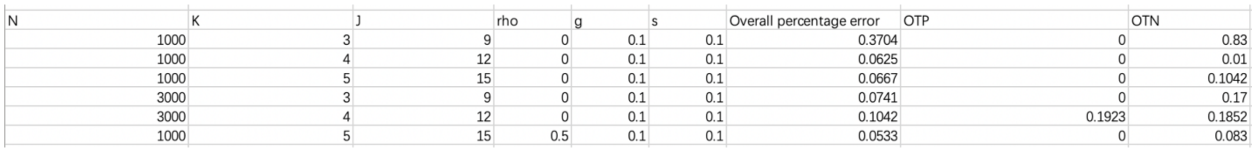
The lamada value ∈{1,2,3,4.5,5,5.5}. The tau value ∈{1,2,2.5,2,3,3.5,5}. The results are shown below.



1	K	J	lamada	tau	g	s	Overall Erro	rho
2	3	9	3	2	0.1	0.1	0.3081	0
3	3	9	3	1	0.1	0.1	0.2348148	0
4	3	9	5	1	0.1	0.1	0.19259	0
5	3	9	4.5	2.5	0.1	0.1	0.2829	0
6	3	9	4.5	2	0.1	0.1	0.26	0
7	3	9	4.5	3	0.1	0.1	0.3081	0
8	3	9	4.5	3.5	0.1	0.1	0.31926	0
9	3	9	4.5	1	0.1	0.1	0.2118519	0
10	3	9	5.5	1	0.1	0.1	0.2177	0
11	4	12	3	2	0.1	0.1	0.4558	0
12	4	12	4.5	2	0.1	0.1	0.37708	0
13	4	12	5	2	0.1	0.1	0.35958	0
14	5	15	4.5	2.5	0.1	0.1	0.5288	0
15	3	9	3	1	0.1	0.1	0.2059259	0.5
16	3	9	3.5	1	0.1	0.1	0.194	0.5
17	3	9	4.5	2.5	0.1	0.1	0.267	0.5
18	3	9	4.5	2	0.1	0.1	0.2518519	0.5
19	3	9	4.5	5	0.1	0.1	0.29	0.5
20	3	9	4.5	1	0.1	0.1	0.1881481	0.5
21	4	12	4.5	2.5	0.1	0.1	0.3892	0.5
22	4	12	4.5	5	0.1	0.1	0.49208	0.5
23	4	12	4.5	2	0.1	0.1	0.3454167	0.5
24	4	12	1	2	0.1	0.1	0.54083	0.5
25	4	12	2	2	0.1	0.1	0.4754167	0.5
26	4	12	3	2	0.1	0.1	0.4054167	0.5
27	5	15	4.5	2.5	0.1	0.1	0.484	0.5
28	5	15	4.5	5	0.1	0.1	0.6096	0.5
29	5	15	4.5	3.5	0.1	0.1	0.554933	0.5

1) When other variables like K, J, λ, g, s, rho are constant, the higher tau is, the larger the overall error. When other variables like K, J, g, s, tau, rho are constant, the higher λ is, the lower the overall error is. Therefore, as K varies, we can usually get the best result when we have a large λ but a small tau.
2) When other variables keep constant, we can find that as K being larger, the overall error of the best model under TLP regularization goes up.
3) When other variables keep constant, we can find that as rho goes up, the overall error of the best model under TLP regularization goes down.
4) We can see TLP performs worse than Lasso when I increase the number of repetitions.

RBM:



N	K	J	rho	g	s	Overall percentage error	OTP	OTN
1000	3	9	0	0.1	0.1	0.3704	0	0.83
1000	4	12	0	0.1	0.1	0.0625	0	0.01
1000	5	15	0	0.1	0.1	0.0667	0	0.1042
3000	3	9	0	0.1	0.1	0.0741	0	0.17
3000	4	12	0	0.1	0.1	0.1042	0.1923	0.1852
1000	5	15	0.5	0.1	0.1	0.0533	0	0.083

1) From the table, we can see that overall true negative percentage error is usually higher compared with the corresponding overall true positive error.
2) When we keep other variables constant, as K increases, the overall percentage error of that model goes down first and then goes up.
3) When we keep other variables constant, as rho increases, the overall percentage error of that model goes down.

Conclusions:

I find some super interesting patterns after doing some experiments. Firstly, when we increase the correlation between different attributes, the overall percentage error goes down. This trend is found in both methods. This result is a bit counterintuitive. Therefore, I need to do more experiments on this. The second interesting trend is that the overall true negative percentage error is usually larger than the corresponding overall true positive percentage error. This might be caused by the algorithm that I used to recover the Q matrix. I will set the entry of the Q-matrix to be 1 if the related delta value is larger than 0. Therefore, I will set the entry of the Q-matrix to be 1 even when the related delta value is super small. Some improvements are needed to solve this problem. I tried to increase the penalty but this will even increase the error when the penalty is too large. Thirdly, when we compare across these two models, we can see that the CDM package method with lasso regularization can give us the lowest overall percentage error. Since RBM is approximating the CDM model, it is reasonable that it cannot give us results that are as good as the results from the CDM model. Lastly, increasing K cannot give us a decisive trend. I think this makes sense because it depends on the attributes themselves. However, increasing N will give us more information, thus giving us a better result.

Reflections:

During this semester, I was firstly reading some papers about CDM and RBM to get the theoretical background of this topic. After that, I start to use the build-in packages and tune the parameters with cross validation method in order to recover the results on the paper. Unfortunately, my results are not as good as the results on the CDM paper with my codes. After getting the original code from chenchen, I run a small case with one repetition successfully but I am not able to run it on cluster because the original code(with parallelization) cannot be run directly on Greatlakes and I don't have much resources left. I also tried the RBM codes on Dropbox. I find that RBM package do a good job in computing Q matrix. After trying some small cases with a few repetitions, I increase the number of repetitions to get a general trend with the help of Greatlakes. I find that increasing the number of repetitions will give us trends that might be different. I think that the next step could be running code on the CDM paper with clusters and build machine learning model with RBM to see whether we can get an even better result. Another direction that I can try is to change the setting J=3K to a more general case.

Reference

1. Jimmy de la Torre(2009), “DINA Model and Parameter Estimation”, Journal of Educational and behavioral statistics, https://journals.sagepub.com/doi/abs/10.3102/1076998607309474
2. Xu and Shang, “Identifying Latent Structures in Restricted Latent Class Models”, https://par.nsf.gov/servlets/purl/10049844
3. Shi(2021), “Cognitively Diagnostic Analysis Using the G-DINA Model in R”, MDPI, https://www.mdpi.com/2624-8611/3/4/52/htm
4. Li, Ma and Xu(2021), “Learning Large Q-matrix by Restricted Boltzmann Machines”, https://arxiv.org/pdf/2006.15424.pdf

Appendix

1. All the codes are saved at: https://www.dropbox.com/home/USPS/Weihan/Final%20Code