POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Q-matrix Refinement, Design and Derivation

PENG XU
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Juillet 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Q-matrix Refinement, Design and Derivation

présentée par **Peng XU**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

John MULLINS, président
Michel DESMARAIS, membre et directeur de recherche
Jean-Marc ROBERT, membre
François BOUCHET, membre externe

DEDICATION

To my beloved family

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor Michel Desmarais, who offered generous help, patience and understanding during my years of Ph.D. research.

I would also like to thank professor Yunxiao Chen for offering me the source code of LASSO algorithm for Q-matrix derivation, and his insight of DINA model.

I could not have completed my study without the support from my family and friends, who never stopped encouraging me to strive for success.

RÉSUMÉ

La Q-matrice établit la correspondance entre les items et les compétences d'un étudiant. Elle revêt une grande importance pour l'évaluation en éducation, mais sa définition et sa validation demeurent un défi, car les véritables relations items-compétences ne peuvent jamais être observées. Les sujets de recherche de cette thèse tournent autour de ce défi, notamment le raffinement d'une Q-matrice définie par un expert, la conception de la Q-matrice la plus efficace, de même que la dérivation de la Q-matrice pilotée par les données. Grâce à l'apprentissage automatique, nous disposons maintenant d'outils puissants pour résoudre ces problèmes.

Nous passons d'abord en revue les modèles et algorithmes associés utilisés dans notre recherche, puis consacrons des chapitres distincts à trois sujets et contributions distinctes.

Le premier sujet est le problème de l'affinement de la Q-matrice, c'est-à-dire la recherche d'une correspondance plus précise entre les items et les compétences que celle prédéfinie. Après avoir d'abord passé en revue trois techniques de raffinement de matrices-Q, à savoir maxDiff, minRSS et ALS, nous proposons ensuite d'utiliser un arbre de décision pour combiner leurs résultats afin d'améliorer leur performance individuelle. Enfin, nous montrons qu'avec la technique de boosting, nous pouvons encore augmenter les performances de l'arbre de décision.

Le deuxième sujet concerne le problème de conception de Q-matrice, qui consiste à trouver une Q-matrice capable de diagnostiquer le plus efficacement possible les profils des étudiants. Nous montrons d'abord que le principe d'identifiabilité est un facteur de qualité déterminant et qu'une Q-matrice non conforme à ce principe ne permet pas d'évaluer correctement les performances des étudiants. Ensuite, les exigences théoriques pour les identifiabilités de paramètres dans le modèle DINA sont passées en revue, car elles servent de guide pour notre stratégie de conception de Q-matrice. Par la suite, nous avons effectué deux expériences pour comparer différentes stratégies de conception de Q-matrice. Le résultat montre que la meilleure stratégie consiste à répéter les vecteurs unitaires lors de la construction d'une Q-matrice. Enfin, une argumentation théorique est avancée pour valider notre revendication de la meilleure stratégie de conception de Q-matrice.

Le dernier sujet est le problème de dérivation de la Q-matrice à partir des données de performance observées des étudiants, sans l'aide d'une Q-matrice prédéfinie. Les techniques de Hill Climbing, LASSO, NMF et ALS sont passées en revue, puis nous proposons notre nouvelle méthode non paramétrique, IRPtoQ, qui utilise le résultat d'un algorithme de clustering

(agglomération) et interprète les centres de clusters comme des modèles de réponse idéaux. Les performances d'IRPtoQ sont comparées à d'autres algorithmes dans différents contextes d'expérimentation. Les résultats montrent que IRPtoQ surpasse les autres méthodes dans la plupart des cas.

ABSTRACT

Q-matrix as a tool for mapping items to skills is of great importance in educational assessment. However, a Q-matrix is merely a way to manually define the mapping between item and skills, since the genuine relationships between them can never be observed. Therefore, the refinement of an expert-given Q-matrix, the design of the most efficient Q-matrix, and the data-driven derivation of the Q-matrix are the research topics of this thesis. With the help of machine learning, now we have more tools to tackle these problems.

We first review the related models and algorithms used in our research and then devote separate chapters to each topic.

The first topic is the Q-matrix refinement problem, that is, to find a more accurate mapping between items and skills than pre-given one. We first reviewed three existing techniques, which are maxDiff, minRSS and ALS respectively. Then we propose to use a decision tree to combine their results in order to achieve better performance. Finally, we show that with boosting technique applied, we can even further increase the performance of the decision tree.

The second topic is the Q-matrix design problem, which is to find a Q-matrix being able to diagnose student profiles the most efficiently. We first show an example of an unqualified Q-matrix, which fails to correctly assess student performance. Then the theoretical requirements for parameter identifiability in the DINA model are reviewed, as they are the guidance for our Q-matrix design strategy. Subsequently, we run two experiments to compare different Q-matrix design strategies, and the results show that the best strategy is to repeat unit vectors in building a Q-matrix. Finally, a theoretical discussion is offered to validate our claim for the best Q-matrix design strategy.

And the last topic is the Q-matrix derivation problem, which is to derive a Q-matrix directly from the observed student performance data, without aid of a pre-given Q-matrix. Techniques of Hill Climbing, LASSO, NMF and ALS are reviewed, then we propose our new nonparameter method, IRPtoQ, which utilizes the result of a clustering algorithm, and interprets the cluster centers as ideal response patterns. The performance of IRPtoQ is compared with other algorithms under different experiment settings. Results show that IRPtoQ is outperforming other methods in most cases.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

RÉSUMÉ v

ABSTRACT vii

TABLE OF CONTENTS viii

LIST OF TABLES x

LIST OF FIGURES xiii

LIST OF SYMBOLS AND ACRONYMS xiv

LIST OF APPENDICES xv

CHAPTER 1 INTRODUCTION 1
- 1.1 Problem definition 1
- 1.2 Thesis Vocabulary 2
- 1.3 Research Questions 3
- 1.4 Contribution 4
  - 1.4.1 First Study: Q-matrix Refinement 4
  - 1.4.2 Second Study: Q-matrix Design 4
  - 1.4.3 Third Study: Q-matrix Derivation 4
- 1.5 Organization of the thesis 4

CHAPTER 2 LITERATURE REVIEW 6
- 2.1 IRT models 6
- 2.2 Latent Class Models 8
  - 2.2.1 Unrestricted Latent Class Model 8
  - 2.2.2 Restricted Latent Class Model 9
- 2.3 Parameter Estimation of DINA model 11
  - 2.3.1 EM 12
  - 2.3.2 MCMC 12

2.4 Nonparametric Method 13

CHAPTER 3 Q-MATRIX REFINEMENT 17
3.1 Introduction 17
3.2 maxDiff 17
3.3 minRSS 20
3.4 ALS 21
3.5 Decision Tree 22
3.6 Boosted Decision Tree 25

CHAPTER 4 Q-MATRIX DESIGN 31
4.1 Introduction 31
4.2 Deterministic Case 32
4.3 Identifiability 34
4.4 Experiment 1: Comparison of three Q-matrix design strategies 35
4.5 Experiment 2: Find best configuration 37
4.6 Theoretical Discussion 42

CHAPTER 5 Q-MATRIX DERIVATION 45
5.1 Introduction 45
5.2 Hill Climbing 45
5.3 LASSO 48
5.4 NMF 50
5.5 ALS 51
5.6 From Clustering to Q-matrix 51
5.6.1 Clustering 51
5.6.2 IRPtoQ: A Nonparametric Method to derive Q-matrix 59

CHAPTER 6 CONCLUSION 69
6.1 Summary of Works 69
6.2 Limitations and future research 69

REFERENCES 71

APPENDICES 79

LIST OF TABLES

Table 2.1	Taxonomy of latent variable models	6
Table 3.1	A toy example to illustrate maxDiff method	18
Table 3.2	Relevant probabilities of correct response based on the δ-method	19
Table 3.3	Q-matrix for refinement	27
Table 3.4	Refinement results for synthetic data	28
Table 3.5	Refinement results for real data	29
Table 5.1	Profile-IRP correspondence example	59
Table 5.2	IRPs derived from clustering algorithm	61
Table A.1	RMSE of clustering, k=3, N=100, s=g=0.05	79
Table A.2	F-score of clustering, k=3, N=100, s=g=0.05	79
Table A.3	Cell-wise accuracy of clustering, k=3, N=100, s=g=0.05	79
Table A.4	Vector-wise accuracy of clustering, k=3, N=100, s=g=0.05	80
Table A.5	RMSE of clustering, N=100, slip=guess=0.1	80
Table A.6	F-score of clustering, k=3, N=100, s=g=0.1	80
Table A.7	Cell-wise accuracy of clustering, k=3, N=100, s=g=0.1	81
Table A.8	Vector-wise accuracy of clustering, k=3, N=100, s=g=0.1	81
Table A.9	RMSE of clustering, k=3, N=100, s=g=0.2	81
Table A.10	F-score of clustering, k=3, N=100, s=g=0.2	82
Table A.11	Cell-wise accuracy of clustering, k=3, N=100, s=g=0.2	82
Table A.12	Vector-wise accuracy of clustering, N=100, s=g=0.2	82
Table A.13	RMSE of clustering, k=3, N=500, s=g=0.05	83
Table A.14	F-score of clustering, k=3, N=500, s=g=0.05	83
Table A.15	Cell-wise accuracy of clustering, k=3, N=500, s=g=0.05	83
Table A.16	Vector-wise accuracy of clustering, k=3, N=500, s=g=0.05	84
Table A.17	RMSE of clustering, k=3, N=500, s=g=0.1	84
Table A.18	F-score of clustering, k=3, N=500, s=g=0.1	84
Table A.19	Cell-wise accuracy of clustering, k=3, N=500, s=g=0.1	85
Table A.20	Vector-wise accuracy of clustering, k=3, N=500, s=g=0.1	85
Table A.21	RMSE of clustering, k=3, N=500, s=g=0.2	85
Table A.22	F-score of clustering, k=3, N=500, s=g=0.2	86
Table A.23	Cell-wise accuracy of clustering, k=3, N=500, s=g=0.2	86
Table A.24	Vector-wise accuracy of clustering, k=3, N=500, s=g=0.2	86
Table A.25	RMSE of clustering, k=4, N=200, s=g=0.05	87

Table A.26 F-score of clustering, k=4, N=200, s=g=0.05 87
Table A.27 Cell-wise accuracy of clustering, k=4, N=200, s=g=0.05 88
Table A.28 Vector-wise accuracy of clustering, k=4, N=200, s=g=0.05 . . 88
Table A.29 RMSE of clustering, k=4, N=200, s=g=0.1 89
Table A.30 F-score of clustering, k=4, N=200, s=g=0.1 89
Table A.31 Cell-wise accuracy of clustering, k=4, N=200, s=g=0.1 90
Table A.32 Vector-wise accuracy of clustering, k=4, N=200, s=g=0.1 . . . 90
Table A.33 RMSE of clustering, k=4, N=200, s=g=0.2 91
Table A.34 F-score of clustering, k=4, N=200, s=g=0.2 91
Table A.35 Cell-wise accuracy of clustering, k=4, N=200, s=g=0.2 92
Table A.36 Vector-wise accuracy of clustering, k=4, N=200, s=g=0.2 . . . 92
Table A.37 RMSE of clustering, k=4, N=500, s=g=0.05 93
Table A.38 F-score of clustering, k=4, N=500, s=g=0.05 93
Table A.39 Cell-wise accuracy of clustering, k=4, N=500, s=g=0.05 94
Table A.40 Vector-wise accuracy of clustering, k=4, N=500, s=g=0.05 . . 94
Table A.41 RMSE of clustering, k=4, N=500, s=g=0.1 95
Table A.42 F-score of clustering, k=4, N=500, s=g=0.1 95
Table A.43 Cell-wise accuracy of clustering, k=4, N=500, s=g=0.1 96
Table A.44 Vector-wise accuracy of clustering, k=4, N=500, s=g=0.1 . . . 96
Table A.45 RMSE of clustering, k=4, N=500, s=g=0.2 97
Table A.46 F-score of clustering, k=4, N=500, s=g=0.2 97
Table A.47 Cell-wise accuracy of clustering, k=4, N=500, s=g=0.2 98
Table A.48 Vector-wise accuracy of clustering, k=4, N=500, s=g=0.2 . . . 98
Table B.1 F-score of derivation, k=3, N=100, s=g=0.05 99
Table B.2 Cell-wise accuracy of derivation, k=3, N=100, s=g=0.05 99
Table B.3 Vector-wise accuracy of derivation, k=3, N=100, s=g=0.05 . . 99
Table B.4 F-score of derivation, k=3, N=100, s=g=0.1 100
Table B.5 Cell-wise accuracy of derivation, k=3, N=100, s=g=0.1 100
Table B.6 Vector-wise accuracy of derivation, k=3, N=100, s=g=0.1 . . . 100
Table B.7 F-score of derivation, k=3, N=100, s=g=0.2 101
Table B.8 Cell-wise accuracy of derivation, k=3, N=100, s=g=0.2 101
Table B.9 Vector-wise accuracy of derivation, k=3, N=100, s=g=0.2 . . . 101
Table B.10 F-score of derivation, k=3, N=500, s=g=0.05 102
Table B.11 Cell-wise accuracy of derivation, k=3, N=500, s=g=0.05 102
Table B.12 Vector-wise accuracy of derivation, k=3, N=500, s=g=0.05 . . 102
Table B.13 F-score of derivation, k=3, N=500, s=g=0.1 103

Table B.14 Cell-wise accuracy of derivation, k=3, N=500, s=g=0.1 103
Table B.15 Vector-wise accuracy of derivation, k=3, N=500, s=g=0.1 . . . 103
Table B.16 F-score of derivation, k=3, N=500, s=g=0.2 104
Table B.17 Cell-wise accuracy of derivation, k=3, N=500, s=g=0.2 104
Table B.18 Vector-wise accuracy of derivation, k=3, N=500, s=g=0.2 . . . 104
Table B.19 F-score of derivation, k=4, N=200, s=g=0.05 105
Table B.20 Cell-wise accuracy of derivation, k=4, N=200, s=g=0.05 105
Table B.21 Vector-wise accuracy of derivation, k=4, N=200, s=g=0.05 . . 106
Table B.22 F-score of derivation, k=4, N=200, s=g=0.1 106
Table B.23 Cell-wise accuracy of derivation, k=4, N=200, s=g=0.1 107
Table B.24 Vector-wise accuracy of derivation, k=4, N=200, s=g=0.1 . . . 107
Table B.25 F-score of derivation, k=4, N=200, s=g=0.2 108
Table B.26 Cell-wise accuracy of derivation, k=4, N=200, s=g=0.2 108
Table B.27 Vector-wise accuracy of derivation, k=4, N=200, s=g=0.2 . . . 109
Table B.28 F-score of derivation, k=4, N=500, s=g=0.05 109
Table B.29 Cell-wise accuracy of derivation, k=4, N=500, s=g=0.05 110
Table B.30 Vector-wise accuracy of derivation, k=4, N=500, s=g=0.05 . . 110
Table B.31 F-score of derivation, k=4, N=500, s=g=0.1 111
Table B.32 Cell-wise accuracy of derivation, k=4, N=500, s=g=0.1 111
Table B.33 Vector-wise accuracy of derivation, k=4, N=500, s=g=0.1 . . . 112
Table B.34 F-score of derivation, k=4, N=500, s=g=0.2 112
Table B.35 Cell-wise accuracy of derivation, k=4, N=500, s=g=0.2 113
Table B.36 Vector-wise accuracy of derivation, k=4, N=500, s=g=0.2 . . . 113

LIST OF FIGURES

Figure 4.1	Q-matrix design strategies	38
Figure 4.2	Three Strategy Comparison on 3-skill case	39
Figure 4.3	Three Strategy Comparison on 4-skill case	39
Figure 4.4	3-skill case, slip=guess=0.01, J=4	40
Figure 4.5	3-skill case, slip=guess=0.2, J=4	40
Figure 4.6	3-skill case, slip=guess=0.01, J=8	40
Figure 4.7	3-skill case, slip=guess=0.3, J=8	40
Figure 4.8	4-skill case, slip=guess=0.01, J=5	41
Figure 4.9	4-skill case, slip=guess=0.2, J=5	41
Figure 5.1	Clustering Method Comparison on k=3, N=100	55
Figure 5.2	Clustering Method Comparison on k=3, N=500	56
Figure 5.3	Clustering Method Comparison on k=4, N=200	57
Figure 5.4	Clustering Method Comparison on k=4, N=500	58
Figure 5.5	Derivation Method Comparison on k=3, N=100	65
Figure 5.6	Derivation Method Comparison on k=3, N=500	66
Figure 5.7	Derivation Method Comparison on k=4, N=200	67
Figure 5.8	Derivation Method Comparison on k=4, N=500	68

LIST OF SYMBOLS AND ACRONYMS

Symbols

N	Number of students
J	Number of items/questions
K	Number of latent skills/attributes
$\boldsymbol{\theta}$	Parameters of a statistical model
s_j	Slip parameter for item j
g_j	Guess parameter for item j
X_{ij}	The real response of student i to item j
ξ_{ij}	The ideal response of student i to item j
$\boldsymbol{\alpha}_i$	Profile vector of student i in Q-matrix based models
η_i	ability parameter of student i in IRT based models

Acronyms

ALS	Alternate Least Square
CDM	Cognitive Diagnosis Model
DCM	Diagnostic Classification Model
DINA	Deterministic Input Noisy And
DINO	Deterministic Input Noisy Or
EM	Expectation Maximization
IRF	Item Response Function
IRT	Item Response Theory
LASSO	Least Absolute Shrinkage and Selection Operator
LVM	Latent Variable Model
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
MLE	Maximum Likelihood Estimation
MM	Mixutre Model
MMB	Mixture of Multivariate Bernoulli
NMF	Nonnegative Matrix Factorization
RMSE	Root Mean Square Error

LIST OF APPENDICES

Appendix A EXPERIMENT RESULTS of CLUSTERING 79

Appendix B EXPERIMENT RESULTS of Q-MATRIX DERIVATION 99

CHAPTER 1 INTRODUCTION

1.1 Problem definition

Suppose an engineering school has a one hour, thirty questions test that is intended to assess mastery of college math for newly enrolled students. College math is, somewhat arbitrarily, broken down into nine topics. How does one decide which question item assesses each of the nine topics? Or suppose a psychologist wants to measure psychological traits of an individual, then how to obtain useful information on these traits from a given questionnaire?

These are both measurement problems. But unlike measuring the height or weight of a man with the help of devices, in the problems above, we are trying to measure unobservable quantities, or say, some latent variables. Since we cannot directly observe them, we have to make inferences based on some observable data, which carries the information of the latent variables we want to measure.

These kinds of problems are typical in psychology and education assessment. The quantities of interest are almost always unobserved. However, there are also some differences in the above scenarios. In the math test case, the latent variable is discrete and usually multidimensional since we are considering more than one skill, while in the psychological case, the latent variable is usually continuous and one-dimensional, since we are concerned with a particular trait.

This thesis is centered around the first scenario, that is, to assess student mastery levels of multiple skills. To tackle this, researchers often predefine an item-skill mapping, which is denoted by a matrix, named Q-matrix (Tatsuoka, 1983). However, this Q-matrix can be misspecified, or misdesigned or difficult to obtain. Therefore, the Q-matrix refinement, design and derivation problems become essential, which are the topics of this thesis. The tools we use to solve these problems are from machine learning and statistics.

1.2 Thesis Vocabulary

In this thesis, we use a collection of terms across all chapters.

`assessment`:	The process of diagnosing student profile of skills
`attribute`:	The ability required to answer a question
`cognitive diagnosis`:	Synonym of assessment
`item`:	A question used in a test
`profile`:	The mastery level of all concerned skills of a student
`respondent`:	The people who participate in tests and skills of whom we want to assess
`skill`:	Synonym of attribute
`student`:	Synonym of respondent

To better illustrate the term attribute and skill, we show an example from (De La Torre, 2009). In the article, the data analyzed is the fraction subtraction data (Tatsuoka, 1990). The author has defined five skills/attributes for solving the fraction subtraction problems which are

1. Subtract basic fractions
2. Reduce and simplify
3. Separate whole from fraction
4. Borrow from whole
5. Convert whole to fraction

We can see that the definitions of skills depend on the domain knowledge, and different experts can have different specifications of them.

Typically, the collected data from students and with which we determine their respective profile can be written in a matrix form. We call it a response matrix or performance matrix. Denote this matrix by R, then R has the form:

$$R = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{array} \begin{array}{c} \begin{array}{ccccccccc} i_1 & i_2 & i_3 & i_4 & i_5 & i_6 & i_7 & i_8 & i_9 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \tag{1.1}$$

in which the rows represents the respondents, or students, while the columns indicate the items, or questions. The letter r stands for respondent and i stands for item. The entries show the observed data, and $R_{ij} = 1$ means the $i-th$ respondent answers the $j-th$ item correctly and $R_{ij} = 0$ means otherwise. An example of Q-matrix is

$$Q = \begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ i_7 \\ i_8 \\ i_9 \end{array} \begin{array}{c} \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array} \tag{1.2}$$

where the letter s stands for skill, and an example of profile matrix is

$$P = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{array} \begin{array}{c} \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array} \tag{1.3}$$

1.3 Research Questions

The research questions investigated in this thesis are:

1. Given a prespecified Q-matrix, how can we improve it based on observed data? Considering there are some off-the-shelf methods, can we combine them to produce better results?

2. How do we design a Q-matrix so that the assessment of skills is more efficient?

3. Can we derive the Q-matrix directly from data, instead of relying on experts? More specifically, can we develop a nonparametric algorithm for this purpose since there are already parametric methods?

1.4 Contribution

1.4.1 First Study: Q-matrix Refinement

In this study, we explore ways to enhance an expert given Q-matrix. In fact, expert can make mistakes in mapping item to skills, and misspecified mapping can cause problems in student profile diagnosis (Rupp & Templin, 2008). Therefore, the research question is, given a possibly misspecified Q-matrix, how can we fix the erroneous values in it based on observed performance data? We propose an improvement over a decision tree approach to combine methods, developed in our lab, and use boosting to further increase the performance of the method, which combines maxDiff, minRSS and ALS methods. The experiments showed that boosting can indeed yield better performance. Section 3 is dedicated to this study.

1.4.2 Second Study: Q-matrix Design

We compared different strategies for Q-matrix design and investigated their relations with parameter identifiability. We claim that the strategy of repeating unit q-vectors is the best way to construct a Q-matrix. Section 4 is dedicated to this study.

1.4.3 Third Study: Q-matrix Derivation

After a review of the heuristic hill climbing, classic LASSO, NMF and ALS methods for Q-matrix estimation, we propose a novel nonparametric way to derive Q-matrix, which is based on the result of clustering performance data and interpret the cluster centers as ideal response patterns. Section 5 is dedicated to this study.

1.5 Organization of the thesis

First, we review the common models in cognitive diagnosis in Chapter 2. Then we discuss the three projects in subsequent chapters. Q-matrix refinement is researched in Chapter 3, Q-matrix design problem is explored in Chapter 4, and Q-matrix derivation problem in Chapter

5. Finally, the work is concluded in Chapter 6. Since the experiment results of Chapter 5 are large, we show figures in Chapter 5, but details of the results are in Appendix A and B.

CHAPTER 2 LITERATURE REVIEW

As discussed in Chapter 1, we want to measure the skill mastery level of students. In order to do so, we need to establish some models and the quantity of interest should be some variable in the model. Moreover, the data we can observe are also in the model as a variable that is dependent on the latent variable we want to infer.

Fortunately, our predecessors have already established a set of interesting models. We will review some of them and emphasize one particular model that our research is based on, the DINA model.

Since we are dealing with *latent variable models*, first we introduce a taxonomy of them. Depending on whether the observed and latent variables are continuous or discrete, we can classify latent variable models into four categories, which is detailed in Table 2.1 (Bartholomew, Knott, & Moustaki, 2011).

Table 2.1 Taxonomy of latent variable models

Observed Variable / Latent Variable	Continuous	Discrete
Continuous	Factor Analysis	Item Response Theory
Discrete	Latent Profile Analysis	Latent Class Model

The response data (observed variable) we are considering here are binary, thus the models we use are the two models in the rightmost column. One is the *item response theory* model and its variants, which usually assume the latent variable to be continuous and the quantity it represents is usually called *trait* or *ability* in the literature. And the other type of models is the *latent class model*, which is also called *diagnostic classification model* in some literature (Rupp, Templin, & Henson, 2010), since respondents are classified into a finite number of groups based on their diagnostic results.

2.1 IRT models

An IRT model is essentially a logistic regression model, but with latent variables. It is a widely researched model and has a huge literature base (Andrich, 1978, 1988; Lord & Novick, 2008; Lord, 2012; Embretson & Reise, 2013; Rasch, 1960). In a typical setting, the quantity we are interested is represented by a continuous latent variable, and depending on the number of parameters, we have 1-PL, 2-PL, 3-PL and 4-PL IRT models, where P stands

for parameter and L stands for logistic. In an IRT model, the likelihood function is also called *item response function* or IRF. For the 1-PL model, the item response function, or the probability of a respondent of ability θ to answer the item j correctly, is:

$$p(X = 1|\theta, b_j) = \frac{1}{1 + e^{-(\theta - b_j)}} \tag{2.1}$$

where the parameter b_j represents the difficulty of the item j, and is called *location parameter*. We can see that when $\theta = b_j$ then $p(X = 1|\theta, b_j) = \frac{1}{2}$, indicating that if the respondent has just reached the difficulty level of item j, then one will answer the item correctly with probability 0.5. If $\theta \gg b_j$, then $p(X = 1|\theta, b_j)$ is close to 1, and if $\theta \ll b_j$, then $p(X = 1|\theta, b_j)$ is close to 0, indicating the two extreme cases of the trait level of the respondent. Notice that following the tradition of latent variable model literature, we distinguish the term "variable" and "parameter". In this model, θ is a latent variable where we want to make inference while b_j is a parameter we want to estimate. The 1-PL is also called Rasch model (Rasch, 1960).

1-PL model did not consider that some item can distinguish respondents better, thus we have the 2-PL model, and the item response function is:

$$p(X = 1|\theta, a_j, b_j) = \frac{1}{1 + e^{-a_j(\theta - b_j)}} \tag{2.2}$$

where the new parameter a_j is the item *discrimination parameter* that determines the steepness of the IRF, the higher the number of a_j, the more effective of the item j to distinguish respondents.

However, low-profile respondents can still answer an item correctly due to chance. To consider this situation, we have the 3-PL model where the item response function is:

$$p(X = 1|\theta, a_j, b_j, c_j) = c_j + (1 - c_j)\frac{1}{1 + e^{-a_j(\theta - b_j)}} \tag{2.3}$$

and where the new parameter c_j is the "guessing" parameter, or the *lower asymptote parameter*. It indicates that even if a respondent is very low on the trait, one can still answer the item correctly. For example, if $c_j = 0.1$, then a respondent has at least a probability of around 0.1 to answer the item j correctly no matter how far away the trait deviates from the item location.

Correspondingly, high profile respondents can still make errors. Adding this factor, we have the 4-PL model and the item response function is:

$$p(X = 1|\theta, a_j, b_j, c_j, d_j) = c_j + (d_j - c_j)\frac{1}{1 + e^{-a_j(\theta - b_j)}} \tag{2.4}$$

where the new parameter d_j is the "slipping" parameter, or the *upper asymptote parameter*. It indicates that even if a respondent is very high on the trait, one can still miss the item. For example, if $c_j = 0.1$ and $d_j = 0.9$, then even if a respondent has a trait far higher than the required amount of b_j, one can still only answer the item correctly with a probability of around 0.9. To summarize, the parameter tuple (a_j, b_j, c_j, d_j) characterizes the item j, where a_j controls its discrimination ability, b_j controls the difficulty, c_j controls the lower bound and d_j controls the upper bound. And the latent variable, or the student ability θ interacts with them through the IRF given by Eq (2.4).

The idea of "slip" and "guess" are also used in other models, which we will discuss below.

2.2 Latent Class Models

Instead of directly modeling the measured quantity, we can assume that there is a finite number of student classes, and each student class shares the same attributes. This idea leads to the *latent class model*. For a comparison, in the IRT model, the latent variable is the student trait. Once the trait is known, the item response probability can be calculated based on the logistic likelihood function. By contrast, in latent class model, the latent variable, or a bunch of them together, represents the student class. Once this class is known, the probability of answering an item correctly can also be calculated based on the likelihood function of that class. If there is no assumption about particular attributes required for items, nor assumptions about how the attributes are utilized, then the latent class model is *unrestricted*. In contrast, if there are assumptions on which attributes are needed for which items, and how the attributes are utilized to construct a response, then the latent class model is *restricted* (Chiu, Douglas, & Li, 2009).

2.2.1 Unrestricted Latent Class Model

In the unrestricted latent class model, we have a single latent variable indicating the student class. Suppose the latent variable indicating the class of student i is z_i, and there are J items, then the conditional likelihood is:

$$p(\boldsymbol{x}_i | z_i = k, \boldsymbol{\theta}) = p_k(\boldsymbol{x_i} | \boldsymbol{\theta}_k) \tag{2.5}$$

where $\boldsymbol{x}_i = (x_1, x_2, ..., x_J)$ is the response vector of student i, $p_k(\cdot)$ is the likelihood function for the k-th class, and $\boldsymbol{\theta}_k$ represents the parameters for the k-th class distribution. However, since the variable z_i is not observed, suppose there are K classes, then the marginal likelihood

is:

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k) \tag{2.6}$$

where π_k is the prior for each class. From the formula above, we see that the likelihood under this model is a mixture of several component distributions. This is why this model is also called *mixture model* in the machine learning community.

The response data we are dealing with are binary, thus we can model them by multivariate Bernoulli distribution, that is (Chiu et al., 2009; Murphy, 2012):

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{J} Ber(x_{ij}|\theta_{jk}) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{J} \theta_{jk}^{x_{ij}}(1-\theta_{jk})^{1-x_{ij}} \tag{2.7}$$

where θ_{jk} is the probability of the k-th class to get the j-th item to be 1. The parameters of this model can be learned by EM algorithm. More detailed discussion of this model can be found in (Bartholomew et al., 2011).

We call this model *Mixture of Multivariate Bernoulli* (MMB), sometimes it is also called *Mixture of Multinoulli* (Murphy, 2012). This model has been used in image classification (Juan & Vidal, 2004), text classification (Juan & Vidal, 2002), but in psychometrics or education assessment, it is not researched much because of the parameters' lack of practical interpretation. Nevertheless, we will use this model for a comparison purpose in Chapter 5.

2.2.2 Restricted Latent Class Model

If we assume relationships between items and latent attributes, then we are in fact adding a kind of restrictions to the latent class model (G. Xu & Shang, 2018), and these types of models are usually named *diagnostic classification model* (DCM) (Rupp et al., 2010) or *cognitive diagnosis model* (CDM) (DiBello, Roussos, & Stout, 2006). Such relationships are often given by a $J \times K$ binary matrix, which corresponds to the *Q-matrix* introduced in the previous chapter (Tatsuoka, 1983, 2009). Latent Class Models that are restricted by a Q-matrix are thus called *Q-restricted latent class model* (G. Xu & Shang, 2018). For simplicity, we will use the term *diagnostic classification model* or DCM in this thesis.

Different from the unrestricted latent class model, where each student belongs to a class which is represented by a single variable, in diagnostic classification models, each student has a profile, which is a vector of their mastery of skills. For example, if there are 3 skills considered, then the profile of student i is denoted by $\boldsymbol{\alpha}_i = (\alpha_1, \alpha_2, \alpha_3)$. Each entry of this vector indicates the mastery level of the corresponding skill. For diagnostic classification models, these entries are binary, indicating there are only 2 possible states of skill mastery,

either mastered or not. It is obvious that in this case the number of different classes of students is finite, and for 3-skill case there are exactly $2^3 = 8$ different class of students. For student profile $\boldsymbol{\alpha}_i$, we use $\theta_{j,\boldsymbol{\alpha}_i}$ to denote the positive response probability for item j:

$$\theta_{j,\boldsymbol{\alpha}_i} = p(x_{ij} = 1|\boldsymbol{\alpha}_i) \tag{2.8}$$

Then the different DCMs are defined by different expressions of $\theta_{j,\boldsymbol{\alpha}_i}$.

Besides of the restrictions on the relationships between items and skills, we can also classify different types based on how the skill and item interact. More specifically, whether the skills needed for an item are in *conjunctive*, *disjunctive* or *compensatory* variants. In conjunctive variant, one must master all required skills to get the correct answer, while in disjunctive variant, just mastering one of the required skills is sufficient for positive response, and last in compensatory variant, the more required skills are mastered, the higher probability to answer the item correctly. We will see examples of them below.

DINA model

Deterministic Input Noisy And (DINA) gate model (Tatsuoka, 1995) is one of the most researched models for educational assessment and is identified as a latent class model in (Haertel, 1989), where the latent variable is defined as:

$$\xi_{ij} = \prod_{k=1}^{K} \alpha_{ik}^{q_{jk}} \tag{2.9}$$

in which α_{ik} is the mastery level of skill k by student i, and the profile of student i, i.e. $\boldsymbol{\alpha}_i = (\alpha_{i1}, \alpha_{i2}, ..., \alpha_{iK})$ is also called the *knowledge state* in (Tatsuoka, 1995). q_{jk} is the requirement of skill k by item j. If skill k is required, then $q_{jk} = 1$, otherwise $q_{jk} = 0$. The vector $\boldsymbol{\xi_i} = (\xi_{i1}, \xi_{i2}, ..., \xi_{iJ})$ is called the ideal response of student i. We can see that this is a conjunctive model since only when $\alpha_{ik} \succcurlyeq q_{ik}$ holds for all k would we expect $\xi_{ij} = 1$. However, there are uncertainties in practice and we can define two types of uncertainty for each item. One is *slip*, denoted by s_j for item j. It represents the probability that a student would miss item j even if all required skills are mastered. The other one is *guess*, denoted by g_j for item j. It represents the probability that a student can guess item j correctly even if at least one required skill is not mastered. With these parameters under consideration, the item response function or the likelihood function of DINA model is:

$$p(X_{ij} = 1|\xi_{ij}) = (1 - s_j)^{\xi_{ij}} g_j^{1-\xi_{ij}} \tag{2.10}$$

DINO model

Deterministic Input Noisy Or (DINO) gate model (Templin & Henson, 2006) is similar to the DINA model, but replaces the conjunctive assumption by the disjunctive assumption. In the DINO model, the latent variable becomes:

$$\omega_{ij} = 1 - \prod_{k=1}^{K} (1 - \alpha_{ik})^{q_{jk}} \tag{2.11}$$

where we can see that so long as $\alpha_{ik} = q_{jk} = 1$ holds for at least one k, we would have $\omega_{ij} = 1$. Similar to DINA model, the likelihood function of DINO model is:

$$p(X_{ij} = 1|\omega_{ij}) = (1 - s_j)^{\omega_{ij}} g_j^{1-\omega_{ij}} \tag{2.12}$$

LLM model

The compensatory model is somewhat between the conjunctive and disjunctive models. An example of a compensatory model is the Linear Logistic Model (LLM) (Maris, 1999) which models the performance data by logistic regression model with latent variables (Hagenaars, 1993). In this model (G. Xu, 2017), the likelihood function for profile $\boldsymbol{\alpha}$ is:

$$\text{logit}(\theta_{j,\alpha}) = \beta_{j0} + \Sigma_{k=1}^{K} \beta_{jk} q_{jk} \alpha_{jk} \tag{2.13}$$

or equivalently:

$$\theta_{j,\alpha} = \frac{\exp(\beta_{j0} + \Sigma_{k=1}^{K} \beta_{jk} q_{jk} \alpha_{jk})}{1 + \exp(\beta_{j0} + \Sigma_{k=1}^{K} \beta_{jk} q_{jk} \alpha_{jk})} \tag{2.14}$$

In this model, each skill contributes to the chances of success of an item.

Note that in this thesis, DINA is the model we use, as it is the most common. While it is not shown that the findings geneneralize to the other models, the investigation methodology can be extended to these models.

2.3 Parameter Estimation of DINA model

While we will focus on the DINA model here, parameter estimation for DINA is much like other latent variable models, and EM and MCMC are the two most used methods for this purpose.

2.3.1 EM

EM (Expectation Maximization) (Moon, 1996) is a general algorithm used in statistics and machine learning to deal with latent variable models (Bishop, 2006; Murphy, 2012). In general, we want to maximize the log likelihood of the observed data:

$$\ell(\theta) = \sum_{i=1}^{N} \log p(\boldsymbol{x_i}|\theta) = \sum_{i=1}^{N} \log \left[\sum_{\boldsymbol{z_i}} p(\boldsymbol{x_i}, \boldsymbol{z_i}|\theta) \right] \tag{2.15}$$

However, this is difficult to compute because of the log of the sum. Instead, EM gets around this problem by considering the complete data log likelihood:

$$\ell_c(\theta) = \sum_{i=1}^{N} \log p(\boldsymbol{x_i}, \boldsymbol{z_i}|\theta) \tag{2.16}$$

This also cannot be computed since $\boldsymbol{z_i}$ is unknown. So we compute the expected complete data log likelihood:

$$Q(\theta, \theta^{t-1}) = E[\ell_c(\theta)|D, \theta^{t-1}] \tag{2.17}$$

where t is the current iteration number, E is the expectation operator and D is the data. This can be proven to be a tight lower bound of $\ell(\theta)$ by using Jensen inequality (Murphy, 2012), and by maximizing this bound we indirectly maximize the real likelihood. Details of using EM on DINA model can be found in (De La Torre, 2009).

2.3.2 MCMC

MCMC is another means to tackle the parameter estimation problems in latent variable models (Murphy, 2012). It is based on a Bayesian perspective. From this perspective, model parameters are treated as random variables, and we estimate the posterior distribution function of them. Usually these posterior distributions are difficult to obtain, but we are only interested in its expectations (as parameter estimator), which can be obtained in a Monte Carlo way. Since this distribution is usually high dimensional (multiple parameters to estimate), a common way to use Monte Carlo method is implementing the Markov Chain Monte Carlo (MCMC). The basic idea of MCMC is to construct a Markov Chain, whose stationary distribution is our distribution of interest, which is the posterior distribution here. Then by drawing samples from this Markov Chain, we can perform Monte Carlo integration with respect to the distribution (Murphy, 2012).

This idea can be applied to DINA model. Different ways to implement MCMC on DINA model have been tried by researchers. First with Metropolis-Hasting sampler (MH sampler)

in (De La Torre & Douglas, 2004), then since for DINA model, the posterior distribution for each parameter can be directly calculated, Gibbs sampler is also used (Li, 2008; DeCarlo, 2012; Culpepper, 2015). After considering the identifiability conditions of DINA model (G. Xu & Zhang, 2015), a constrained Gibbs sampler is proposed in (Chen, Culpepper, Chen, & Douglas, 2018). In recent years, the No-U-Turn sampler (NUTS) (Hoffman & Gelman, 2014), an extension of Hamiltonian Monte Carlo (Duane, Kennedy, Pendleton, & Roweth, 1987) is proposed to implement MCMC. Researchers have claimed that it performs better than traditional MH sampler and Gibbs sampler (Nugroho & Morimoto, 2015; Grant, Furr, Carpenter, & Gelman, 2017). Research on using No-U-Turn sampler on DINA model can be found in (da Silva, de Oliveira, von Davier, & Bazán, 2018).

2.4 Nonparametric Method

Before reviewing the nonparametric methods used in the student profile assessment, some remarks on the term *nonparametric* needs to be addressed since this term has different meanings in the statistics and machine learning literature.

On the one hand, it means literally "nonparametric", indicating there are no statistical parameter involved, or in other words, "distribution free" (Kendall, Stuart, Ord, & Arnold, 1999). Examples of this usage include *order statistics*, which uses ranks of samples to conduct statistical inference, and *sign test*, which conducts a statistical test based on the signs of samples. Neither of these methods requires presumptions on the distribution of samples, nor do they involve parameters.

On the other hand, the term "nonparametric" can also mean the structure of a statistical model is not fixed (Hjort, Holmes, Müller, & Walker, 2010). Sometimes, it might be even better to call it "infinite parametric" (Wasserman, 2006). In fact, the data is modeled from a collection of distributions, or distribution of distributions (Murphy, 2012). For example, a Dirichlet process mixture model (Antoniak, 1974; Murphy, 2012) is a mixture model, but it does not assume the number k of mixtures. That means each sample comes from a mixture of clusters, and each cluster is modeled by a Dirichlet distribution. The number of clusters k is not predetermined, and can be any positive integer, thus the weights of mixtures are also not fixed. Usually, a prior is given to these weights to model the process. As the number of samples grows, the number of clusters also increases. For a single discrete latent variable like the number of mixtures k, the most commonly used prior is the *Chinese Restaurant Process* (Aldous, 1985; Pitman, 2002) which implies the idea of "rich get richer", meaning a new sample is more likely to be assigned to a cluster having the most samples, instead of creating a new cluster for it (Murphy, 2012). It is worth mentioning that Dirichlet process

mixture model has also found its application in student performance prediction (Lindsey, Khajah, & Mozer, 2014) and cognitive process modeling (Austerweil, Gershman, Tenenbaum, & Griffiths, 2015). A good tutorial of Bayesian nonparametric models and their applications in cognitive science can be found in (Gershman & Blei, 2012). However, in this thesis, all the nonparametric methods we discussed are using the first meaning of the term nonparametric.

Chiu and Douglas (2013) proposed a nonparametric method to classify students given their response data and Q-matrix. This method links the Q-matrix to *ideal response patterns*. In essence, for a given number of attributes or skills, there are fixed number of profile patterns (e.g, for 3 skills, there are $2^3 = 8$ possible profile patterns). Then, since the Q-matrix is given, for each profile pattern, we can obtain its ideal response pattern. If there are 8 possible profile patterns, then there are also 8 ideal response patterns. To classify the students, we just need to assign each student to the class that has the closest ideal response pattern based on some distance measure.

To illustrate this idea, let us look at an example below. Suppose there are 3 skills, thus the 8 possible profile patterns are:

$$\begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & a_2 & a_3 \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{array}\right] \end{array}$$

Suppose the Q-matrix for 4 items are:

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & a_2 & a_3 \end{array} \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{array}\right] \end{array}$$

then, the corresponding ideal response patterns are:

$$\begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{array} \begin{array}{c} \begin{array}{cccc} q_1 & q_2 & q_3 & q_4 \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right] \end{array}$$

Take p_6 for instance, respondents of profile p_6 master skills/attributes a_1 and a_3, therefore they should succeed in any item that requires only a_1 or a_3 or both, which are items q_1 and q_3 in this example. We see that there is a 1–1 correspondence between the profile patterns and the ideal response patterns. Thus if we know the ideal response pattern of one student, then we know the profile pattern. Therefore, the core problem is to determine the ideal response pattern of each student based on observed noisy responses.

Naturally, we would want to compare the observed responses with all ideal response patterns, and the closest ideal response patterns should be the true ideal response patterns of students. Therefore, choosing the right distance measure to define the closeness is key to the success of the method. Chiu and Douglas (2013) suggest to use the *weighted Hamming Distance*. First we know that Hamming distance is a commonly seen distance measure for binary data. It is defined as:

$$d_h(\boldsymbol{x}, \boldsymbol{\eta}) = \sum_{j=1}^{J} |x_j - \eta_j|$$

where $\boldsymbol{x}$ and $\boldsymbol{\eta}$ denote the observed response vector and the ideal response pattern vector respectively.

However this measure does not take into account the uncertainty of items. For example, if item j_1 is answered correctly by 90% of students while item j_2 is answered correctly by only 50%, then, when calculating distance, item j_1 should have higher weights since it is more reliable. Based on this idea, Chiu and Douglas (2013) proposed the weighted Hamming distance

$$d_{\omega h}(\boldsymbol{x}, \boldsymbol{\eta}) = \sum_{j=1}^{J} \frac{1}{\bar{p_j}(1 - \bar{p_j})} |x_j - \eta_j|$$

where $\bar{p_j}$ is the proportion correct on the jth item.

Besides the uncertainty directly coming from different uncertainty level of items, we can also apply different weights on guessed and slipped responses. For example, if the item is an open-ended question, then it makes sense to penalize guess, while if the item is a multiple-choice question, then at least guess should not be penalized as much as slip since it is perhaps easier to guess correctly than choosing incorrectly while in fact mastered all the required skills. With these considerations, Chiu and Douglas (2013) defined *penalized Hamming distance*

$$d_{gs}(\boldsymbol{x}, \boldsymbol{\eta}) = \sum_{j=1}^{J} \omega_g I[x_j = 1]|x_j - \eta_j| + \sum_{j=1}^{J} \omega_s I[x_j = 0][x_j - \eta_j|$$

where ω_g and ω_s are weights for the guess and slip items respectively, $I[\cdot]$ is the indicator function. When $\omega_g = \omega_s = 1$, this penalized Hamming distance reduces to Hamming distance. In other words, this penalization is targeting at cases when slip and guess are supposed to be different.

There are some advantages to using the nonparametric methods for student classification. First, *efficiency*, the method does not involve much computation compared to other parametric methods. Second, *flexibility*, the method is basically model free, it can be used in models so long as ideal response pattern can be defined through Q-matrix. Third, *robustness*, when the given Q-matrix is misspecified, the nonparametric method can still classify students even better than model-based methods.

CHAPTER 3 Q-MATRIX REFINEMENT

3.1 Introduction

The Q-matrix is often expert-given. However, experts can make mistakes and a misspecified Q-matrix leads to incorrect student skill diagnosis (Rupp & Templin, 2008). Moreover, even if the expert is confident in the Q-matrix, there might still be some space for improvement. These considerations lead to the field of Q-matrix refinement, which differs from the problem of Q-matrix derivation because it starts from a given plausible Q-matrix, unlike the complete unknowingness situation in Q-matrix derivation problems. In general, we assume that this Q-matrix is based on expert knowledge thus it only has small differences from the true Q-matrix. Mathematically, the problem can be formulated as finding a function f, that $Q_{new} = f(R, Q_{old})$, where R is the response matrix and Q_{old} is the original Q-matrix that needs to be refined.

In this chapter, we discuss different Q-matrix refinement techniques, and propose ways to enhance them. In Section 3.2, we introduce the maxDiff method that relies on maximizing the difference between mastered profile and not-mastered profile. In Section 3.3, we introduce a nonparametric method by minimizing a residual of squared sums. In Section 3.4, we talk about a matrix factorization method named ALS. In Section 3.5 we propose to use a decision tree to combine results from previous methods. And finally in Section 3.6, we further enhance the result of a decision tree by using Adaboost algorithm.

3.2 maxDiff

An early method of Q-matrix refinement is maxDiff (De La Torre, 2008)[1]. This method is based on the observation that a correctly specified q-vector should maximize the difference of the probability of answering correctly between respondents who master all required skills and respondents who do not. To see this, we can look at table 3.1.

For the 3-skills case, we can see that there are eight ($2^3 = 8$) possible skill patterns, or profile patterns. η is the ideal response, that is, the response if there is no slip or guess involved. Table 3.1 shows the probabilities of correct response for a hypothetical item in a 3-attribute domain, where attribute 1 and 2 are required to answer the item correctly, meaning the q-vector can be denoted as (1,1,0). The second column shows the ideal response, η. Column 3,

[1]The names maxDiff and minRSS (section 3.3) are from M. C. Desmarais, Xu, and Beheshti (2015), not from the original authors.

Table 3.1 A toy example to illustrate maxDiff method

	Under True Q Vector		Attributes (skill pattern)			Under Pattern Q Vector		
Pattern	η	$P(X=1\|\eta)$	α_1	α_2	α_3	$P(X=1\|\eta^*=0)$	$P(X=1\|\eta^*=1)$	δ^*
1	0	0.20	0	0	0	-	-	-
2	0	0.20	0	0	1	0.35	0.35	0.00
3	0	0.20	0	1	0	0.20	0.50	0.30
4	0	0.20	0	1	1	0.30	0.50	0.20
5	0	0.20	1	0	0	0.20	0.50	0.30
6	0	0.20	1	0	1	0.30	0.50	0.20
7	1	0.80	1	1	0	0.20	0.80	0.60
8	1	0.80	1	1	1	0.29	0.80	0.51

$P(X=1|\eta)$, shows the probability of answering it correctly when slip and guess are involved. Both slip and guess are assumed to be 0.2 in this simple example. For example, in the third row, the skill pattern is $(0,1,0)$, then its ideal response η is 0 since the true q-vector is $(1,1,0)$. Subsequently in this case, $P(X=1|\eta) = guess = 0.2$. We know that only skill patterns 7 and 8 allow a correct answer to this item. Thus in these two rows, $P(X=1|\eta) = 1 - slip = 0.8$.

In practice we do not know the true q-vector is $(1,1,0)$, thus we want to examine all the possible q-vectors (which are the same as all possible skill patterns) and record relevant values. Now we have to use η^* instead of η to represent the possible ideal response since the true ideal response is unknown. The values we are interested now are $P(X=1|\eta^*=0)$, $P(X=1|\eta^*=1)$ and δ^*, where $\delta^* = |P(X=1|\eta^*=1) - P(X=1|\eta^*=0)|$. In the expression of δ^*, the first term is the probability of answering the item correctly by the "mastered" profile patterns, while the second term is the probability of answering the item correctly by the "not-mastered" profile patterns. The idea of maxDiff is that the true q-vector should maximize the difference of these two probabilities. We still use the third row as an example to show how these values are calculated. In this case, the examined q-vector is $(0,1,0)$, thus there are four profile patterns that cover this Q-vector (profiles that leads to $\eta^*=1$), and four profile patterns that do not (that lead to $\eta^*=0$). Among the four profile patterns that cover this Q-vector, two (profiles (1,1,0) and (1,1,1)) will answer correctly with probability 0.8. The other two (profiles (0,1,0), (0,1,1)) will answer correctly with probability 0.2. In the other hand, among the four profile patterns that do not cover this Q-vector, all of them (profiles (0,0,0), (0,0,1), (1,0,0), (1,0,1) will answer correctly with probability 0.2.

Therefore, we have:

$$P(X = 1|\eta^* = 1) = (0.2 * 2 + 0.8 * 2)/4 = 0.50$$

$$P(X = 1|\eta^* = 0) = (0.2 * 4)/4 = 0.20$$

$$\delta^* = P(X = 1|\eta^* = 1) - P(X = 1|\eta^* = 0) = 0.50 - 0.20 = 0.30$$

In the same way we can calculate δ^* for all possible q-vectors and record relevant values in the last three columns. We find that the true q-vector $(1, 1, 0)$ has the largest $\delta^* = 0.60$, which validates the hypothesis of the maxDiff method.

To put maxDiff into practical usage, a straightforward thought is to calculate δ^* for all possible q-vectors, as in table 3.1, however this is computationally expensive with complexity $O(2^K)$. (De La Torre, 2008) proposed a sequential method that reduces the complexity to $O(K^2)$ which is named as δ-method.

Table 3.2 shows how the δ-method is done for a 3-skill q-vector. In the first step, we consider the 1-skill case, the column 3-5 correspondent to profile (1,0,0), (0,1,0) and (0,0,1). We can see that either skill 1 or 2 offers the largest difference 0.30 (0.50-0.20). We can choose either of them as the first skill. Here we choose skill 1. In the the second step, skill 1 is fixed and we need to decide whether to include skill 2 or 3, that is, to check profile (1,1,0) and (1,0,1). We see that profile (1,1,0) gives the largest difference, 0.60, thus we keep skill 2 and move to step 3. For this step, we need to decide whether to include skill 3 by checking profile (1,1,1), and we see that the difference is decreased from 0.60 to 0.51, which implies that skill 3 should not be included. We see that in this δ-method, the number of δ^* has been calculated $(K^2 + K)/2$ times for K-skill domains, which is much less than $2^K - 1$ times in the exhaustive search algorithm.

We can see that the calculation of $P(X = 1|\eta^* = 1)$ and $P(X = 1|\eta^* = 0)$ depends on slip,

Table 3.2 Relevant probabilities of correct response based on the δ-method

		Sequential Steps		
Number of Attributes	η^*	1	2	3
One	0	0.20	0.20	0.35
	1	0.50	0.50	0.35
Two($\alpha^{(1)} = \alpha_1$)	0	-	0.20	0.30
	1	-	0.80	0.50
Three($\alpha^{(1)} = \alpha_1$, $\alpha^{(2)} = \alpha_2$)	0	-	-	0.29
	1	-	-	0.80

guess, and the ratio of each profile pattern, which needs to be estimated given the prespecified Q-matrix. This can be done through an EM algorithm, details can be found in De La Torre (2009).

Based on the above discussion, the maxDiff method involves two steps. First, with a prespecified Q-matrix, use the EM algorithm to estimate slip, guess and profile proportions. Then use these estimated parameters with the δ-method and construct a new Q-matrix. Detailed examples can be found in (De La Torre, 2008).

3.3 minRSS

Another method for Q-matrix validation proposed by Chiu (2013) is a nonparametric method, which we denote as minRSS in this thesis, meaning *minimization of Residual Sum of Squares*.

The nonparametric minRSS approach takes advantage of the nonparametric method of student classification we introduced in section 2. With the pregiven Q-matrix, we can obtain the profile of each student using that method. Then, with these student profiles, we make inferences on each q-vectors. The hypothesis is, if the Q-matrix is correct, then the student response should be as close to the ideal response pattern as possible. Based on this hypothesis, we can calculate the deviance between observed responses and ideal responses for each item and each student, this deviance is named RSS (*Residual Sum of Squares*).

Denote Y_{ij} and η_{ij} as the observed and ideal response of student i to item j, then the RSS of item j for student i is defined as:

$$RSS_{ij} = (X_{ij} - \eta_{ij})^2$$

And the RSS of item j across all students is defined as:

$$RSS_j = \sum_{i=1}^{N}(X_{ij} - \eta_{ij})^2 = \sum_{m=1}^{2^K}\sum_{i \in C_m}(X_{ij} - \eta_{jm})^2$$

And a large RSS_j indicates that the q-vector for item j might not be correct. Thus for each item j, we only need to calculate the RSS_j for all q-vectors, and choose the q-vector that yields the smallest RSS_j to be the refined new q-vector. This updating process is done one by one for each item and finally outputs a refined new Q-matrix. More details can be found in Chiu (2013).

3.4 ALS

Matrix factorization is a common technique used in statistics and machine learning. In fact, for the observed target variable, denoted as $\boldsymbol{y}$, the fitting of linear regression model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon}$ is equivalent to find $\boldsymbol{w}$ that minimize the distance $||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2$, the Ordinary Linear Squares (OLS) method gives the parameter estimation

$$\hat{\boldsymbol{w}}_{OLS} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} = \boldsymbol{X}^{+}\boldsymbol{y}$$

where $\boldsymbol{X}^{+}$ denotes the Moore-Penrose Generalized Inverse of matrix $\boldsymbol{X}$(Penrose, 1956). This result can be extended to matrix case. In fact, if $\boldsymbol{Y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_n})$ is a matrix, each column vector $\boldsymbol{y_i}$ will yield a parameter vector $\boldsymbol{w_i}$ using above formula, then denote $\boldsymbol{W} = (\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_n})$, we have(Penrose, 1956):

$$\boldsymbol{W} = \boldsymbol{X}^{+}\boldsymbol{Y}$$

which minimize the distance $||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W}||_F{}^2$, and thus $\boldsymbol{X}\boldsymbol{W}$ can be seen as an approximate matrix factorization of $\boldsymbol{Y}$.

Suppose that for student response data, we also have the linear interaction between student profile and Q-vectors, then the model is:

$$\boldsymbol{X}_{\boldsymbol{m}\times\boldsymbol{n}} = \boldsymbol{P}_{\boldsymbol{m}\times\boldsymbol{k}}\boldsymbol{Q}^{\boldsymbol{T}}_{\boldsymbol{n}\times\boldsymbol{k}} + \boldsymbol{\epsilon}_{\boldsymbol{m}\times\boldsymbol{n}}$$

where m is the number of students , n is the number of items, k is the number of skills, $\boldsymbol{P}$ is the profile matrix and $\boldsymbol{Q}$ is the Q-matrix. For simplicity, we omit the subscripts, that is:

$$\boldsymbol{X} = \boldsymbol{P}\boldsymbol{Q}^{\boldsymbol{T}} + \boldsymbol{\epsilon}$$

In Q-matrix validation problem, there is a pre-given Q-matrix $\boldsymbol{Q_0}$, thus we can calculate a $\boldsymbol{P_0}$ using the formulas above, that is:

$$\boldsymbol{P_0} = \boldsymbol{X}(\boldsymbol{Q}^{\boldsymbol{T}}_{\boldsymbol{0}})^{+}$$

[2]Here F denotes the Frobenius Norm, note that $||\cdot||_2$ in matrix case is induced by ℓ_2 vector norm and we have the inequality $||\boldsymbol{A}||_2 = \sigma_{max}(\boldsymbol{A}) \leq ||\boldsymbol{A}||_F = (\Sigma_{i=1}^{m}\Sigma_{j=1}^{n}|a_{ij}^2|)^{1/2}$, where $\sigma_{max}(\boldsymbol{A})$ represents the largest singular value of matrix $\boldsymbol{A}$.

with this $\boldsymbol{P_0}$ we can calculate $\boldsymbol{Q}$ again, denoted as $\boldsymbol{Q_1}$, that is:

$$\boldsymbol{Q_1^T} = \boldsymbol{P_0^+ X}$$

then we can calculate $\boldsymbol{P}$ again, this process will generate a sequence of $\boldsymbol{P_0}, \boldsymbol{P_1}, \boldsymbol{P_2}, ...$ and a sequence of $\boldsymbol{Q_0}, \boldsymbol{Q_1}, \boldsymbol{Q_2}, ...$, both of these two sequences are convergent, since each step of calculation is equivalent to minimizing the distance $||\boldsymbol{X} - \boldsymbol{PQ^T}||_F$, and the distance sequence $d_0, d_1, d_2, ...$ is decreasing and always positive, from calculus we know it is convergent, thus the sequence $\{\boldsymbol{P_i}\}$ and sequence $\{\boldsymbol{Q_i}\}$ are also convergent.

M. C. Desmarais and Naceur (2013) proposed to use the above method to obtain the refined Q-matrix $\boldsymbol{Q}_{ref}$. Note that this model is a compensatory model, that is, the more skills a student mastered for an item, the higher probability that item will be answered right. This is not the case of the commonly researched DINA model, which is a conjunctive model, meaning one must master all required skills to answer an item correctly. Fortunately, (M. C. Desmarais, Beheshti, & Naceur, 2012) and (Sun, Ye, Inoue, & Sun, 2014) points out that for the conjunctive model, we have:

$$\boldsymbol{X} = \overline{\overline{\boldsymbol{P}} \odot \boldsymbol{Q^T}}$$

where $\odot$ denotes the Boolean Matrix Product.

Note that this problem is no longer an ordinary matrix factorization problem, but a *Binary Matrix Factorization*(BMF) problem. BMF is much more difficult due to the limitation of matrix entries to be 0 and 1. Sun et al. (2014) also tries to implement BMF with an alternating methods, akin to ALS. However it gets more computationally intensive. Here since the main aim is for Q-matrix validation, M. C. Desmarais et al. (2012) consider using ordinary matrix factorization instead, and evaluate the obtained new Q-matrix directly. To separate this conjunctive case with the compensatory case, this method is denoted as ALSC. By comparison of prediction performance, it is shown that the ALSC proposed new Q-matrix is better than the original given Q-matrix.

3.5 Decision Tree

With those Q-matrix validation methods discussed before, a natural idea is to find a way to combine them to obtain a more powerful tool. In other words, for initial Q-matrix $\boldsymbol{Q^0}$, we use methods 1, 2, 3, etc., to obtain new proposed Q-matrices $\boldsymbol{Q^1}$, $\boldsymbol{Q^2}$, $\boldsymbol{Q^3}$ etc, and we want to find a combined refinement $\boldsymbol{Q}_{combined}$, that is

$$\boldsymbol{Q}^{combined} = f(\boldsymbol{Q^1}, \boldsymbol{Q^2}, \boldsymbol{Q^3}, ...)$$

Thus the problem is how to find a satisfactory function f. Obviously, we can manually set rules for f, for example, this f can be vote-based. For an entry Q_{ij}, if both Q_{ij}^1 and Q_{ij}^2 says it is 1, but Q_{ij}^3 says it is 0, then we decide it is 1 since 1 has more votes than 0. However we know that different refinement methods might have different accuracy and one entry's prediction might not be only dependent on that specific entry. Therefore, can we learn the rules of f from the data instead of setting the rules by ourselves? The answer is yes and a classical approach to do so is *decision tree*.

A decision tree is a well-known technique in machine learning that serves as an *ensemble learning* method to combine different base classifiers or regressers. There are several algorithms to estimate the parameters of decision tree, including ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993) and CART (Breiman, Friedman, Stone, & Olshen, 1984). Except for ID3, these algorithms have two steps (ID3 only has the first step). The first step is to grow the trees, which is to sequentially choosing feature variables by checking their contribution of uncertainty change. For example, denote target variable as Y and feature variables as X_1, X_2, etc., then the first step is to compare $Uncertainty(Y)$ versus $Uncertainty(Y|X_i)$ to obtain the information gain. The uncertainty can be measured by entropy, like in ID3 and C4.5, or Gini Index, like in CART. The uncertainty change can be measured by difference, which is called *information gain* (ID3, CART), or by a ratio, which is called *information gain ratio* (C4.5). The second step is pruning, which is to collapse those nodes that are too deep to avoid overfitting, and is implemented by minimizing a loss function for all nodes as a whole and penalized by number of nodes.

Using terms of machine learning, decision tree is a *supervised learning* method, which means it requires a training dataset to estimate the parameters of the model. In our case, how do we design such a training dataset to learn the rules? M. C. Desmarais et al. (2015) propose to use simulated data and a way to generate it. The whole process starts from Q-matrix Q_0, suppose it is a 9×3 matrix, that is, 9 items and 3 skills, then using it as input, the following steps are implemented

1. Randomly permute the Q-matrix to obtain up to 1000 new Q-matrices Q_1, Q_2, ..., Q_{1000}, this randomization is done cell-wisely, meaning only the ratio of 0 and 1 are kept.

2. For each Q_i, generate a response matrix X_i with 400 respondents, using DINA model, with slip=guess=0.2 and profiles uniformely distributed.

3. For each Q_i, since there are $9 \times 3 = 27$ cells, for 1-cell perturbation, we have 27 possible ways, and these 27 pertubated Q-matrix will be denoted as Q_i^1, Q_i^2,...,Q_i^{27}

4. For each validation method, calculate the refined Q-matrix. For instance, feed X_i and Q_i^1 to minRSS, maxDiff and ALSC method we will yield refined Q-matrices $Q^1_{i_{minRSS}}$, $Q^1_{i_{maxDiff}}$ and $Q^1_{i_{ALSC}}$ respectively.

5. Calculate other factors.
   (1) Skills per Row (SR): The number of skills required of an item
   (2) Skills per Column (SC): Times of the skill get tested across all items
   (3) Stickiness Factor (SF): The rigidity of a cell under each validation method. To get the intuition of this factor, we look at an example for minRSS method. For a 9×3 Q-matrix in a particular run of experiment, there are 27 possible perturbation places. And suppose a particular cell, like cell 1, has 12 false positive (minRSS proposed to change, but it should not change) during the 27 cases, then the stickiness factor of minRSS for this cell is 12/27=0.44.

6. Feed the data into a decision tree.

The final dataset generated before feeding into a decision tree from the above procedure will have the form below

	Algorithm target prediction			Other factors				
Target	minRSS	maxDiff	ALSC	SR	SC	SF.minRSS	SF.maxDiff	SF.ALSC
1	1	0	1	3	5	0.44	0.32	0.27
0	0	1	0	3	7	0.35	0.37	0.52
...	...	...	...	...	...	...	...	...

Other factors are designed to help decision tree using more information to yield better performance. Based on the proposed factors, three different strategies of decision tree are proposed. They are:

(1) minRSS+maxDiff+ALSC

(2) minRSS+maxDiff+ALSC+SR+SC

(3) minRSS+maxDiff+ALSC+SR+SC+Stickiness.minRSS+Stickiness.maxDiff+Stickiness.ALSC

The performance of the decision tree is measured by two metrics, accuracy and F-score respectively. To define them, first we look at the four cases for each cell

- **True Positive (TP)**: perturbed cell that was correctly changed
- **True Negative (TN)**: non perturbed cell left unchanged

- **False Positive (FP)**: non perturbed cell incorrectly changed
- **False Negative (FN)**: perturbed cell left unchanged

Then the accuracy is defined as

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

And the F-score is defined as

$$Fscore = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

where

$$Precision = \frac{TP}{TP + FP}$$

and

$$Recall = \frac{TP}{TP + FN}$$

Experiments in (M. C. Desmarais et al., 2015) showed that the strategy 3 has the best performance, thus it is used in our next Q-matrix refined method, *boosted decision tree*. We will look at the detailed results when we compare them with the Boosted decision tree ones in table 3.4.

3.6 Boosted Decision Tree

Boosting is a popular technique in Machine Learning. It transforms a weak learner into a strong one by reweighting the training samples and combining multiple learners. The word "weak" means it performs only better than random guessing. Originally boosting was designed for two-class classification (Schapire, 1990), later it was generalized to multi-class classification (J. Friedman, Hastie, & Tibshirani, 2000; Hastie, Rosset, Zhu, & Zou, 2009) and regression (J. H. Friedman, 2002).

Boosting was first proposed in (Schapire, 1990) in the PAC-learning framework (Valiant, 1984; Kearns, Vazirani, & Vazirani, 1994). In this early version of boosting, the implementation follows the steps below. First an initial classifier h_1 is trained on N samples, then a second classifier h_2 is trained on a new group of N samples, half of which are misclassified by h_1. Subsequently, h_3 is trained on N samples where h_1 and h_2 disagree. And the final boosted classifier is $h_B = \text{MajorityVote}(h_1, h_2, h_3)$.

Schapire's work was further advanced in (Freund, 1995) by combining many weak learners

simultaneously. However, both these two algorithms require producing a classifier with a fixed error rate, this limitation was removed in the later development in Freund and Schapire (1996a) which is adaptive and named *AdaBoost*. This algorithm became widely spread and praised. In fact, this algorithm was so successful that it won the prestigious Gödel prize. It is also used in our experiment for Q-matrix refinement.

Many researchers tried to offer theoretical explanation of boosting. An overview can be found in Meir and Rätsch (2003). Originally, using PAC learning, (Freund & Schapire, 1996a) and (Schapire & Singer, 1999) offers upper bounds on generalization error to offer support for their algorithms, but those bounds are too loose to explain the high performance of boosting. Other explanations come from game theory (Freund & Schapire, 1996b), VC theory (Schapire & Singer, 1999), additive models (J. Friedman et al., 2000) and dynamic systems (Rudin, Daubechies, & Schapire, 2004).

While Boosting can be used with any classifier, many authors have explored it with decision tree. In fact, Adaboost with decision trees are even called the "best off-the-shelf classifier in the world" (Breiman, 1999). In this thesis, the decision tree type we used is *CART* which was introduced in the previous section, since it is the most popular one due to its availability of handling missing data and mixed inputs with both categorical and continuous data (Murphy, 2012).

For a training set of N samples, the whole procedure for Adaboost is shown below (Freund & Schapire, 1996a; Murphy, 2012):

Algorithm 1: Adaboost

---

1. Start with weights $\omega_i = 1/N, i = 1, ..., N$;
2. **for** $i \leftarrow 1$ **to** M **do**
   - (a) Fit the classifier $\phi_m(x) \in \{-1, 1\}$ using weights ω_i on the training data;
   - (b) Compute $err_m = \dfrac{\sum_{i=1}^{N} \omega_i I(\tilde{y}_i \neq \phi_m(x_i))}{\sum_{i=1}^{N} \omega_i}$;
   - (c) Compute $\alpha_m = \log[(1 - err_m)/err_m]$;
   - (d) Set $\omega_i \leftarrow \omega_i \exp[\alpha_m I(\tilde{y}_i \neq \phi_m(x_i)]$ and renormalize so that $\sum_i \omega_i = 1$;

   end
3. Output the classifier

$$f(x) = \text{sgn}(\sum_{m=1}^{M} \alpha_m \phi_m(x))$$

In which M is the number of iterations (10 in our experiment), ω_i, is the weight for i-th data,

$I(\cdot)$ is the indicator function, $\tilde{y}_i \in \{-1, 1\}$ is the class label of training data, and $\phi_m(x)$ is the decision tree model in our case.

We apply this boosting technique on our decision tree model introduced in the previous section. Based on the result of the comparison of the three strategies, we implement the best performing one, that is, besides using the outputs of three different refinement methods (minRSS, maxDiff and ALSC), we also use extra features including skills per row, skills per column and stickiness for each classifier.

In order to compare with results obtained in the previous decision tree method, we use the same procedure to generate synthetic datasets and also compare on the same real datasets. The real dataset we are considering is the famous fraction algebra dataset (Tatsuoka, 1984). A number of Q-matrices have been proposed for this Q-matrix, three of them are used in our research together with a Q-matrix proposed by ourselves. Table 3.3 shows these Q-matrices we used.

To evaluate the performance, we use the same two metrics introduced in the previous section, i.e, the accuracy and F-score.

The results of our study (P. Xu & Desmarais, 2016) show that the decision tree strategy can be further improved with boosting. At the synthetic data side, the results show an F-score error reduction gain from boosting over the DT score of close to 50% on average for all four Q-matrices (Table 3.4), while a 17.8% reduction for real data (Table 3.5). The error reduction gain of F-scores from F_0 to F_1 is computed as follow:

$$\text{Gain}(F_0, F_1) = \frac{F_1 - F_0}{1 - F_0}$$

Table 3.3 Q-matrix for refinement

Name	**Number of**			**Description**
	Skills	Items	Cases	
QM1	3	11	536	Expert driven from (Henson, Templin, & Willse, 2009)
QM2	3	11	536	Expert driven from (De La Torre, 2008)
QM3	5	11	536	Expert driven from (Robitzsch, Kiefer, George, & Ünlü, 2017)
QM4	3	11	536	Data driven, SVD based

Table 3.4 Refinement results for synthetic data

	Individual			Ensemble			
QM	**minRSS**	**maxDiff**	**ALSC**	**DT**	**Gain%**	**BDT**	**Gain%**
Accuracy of perturbated cells							
1	0.809	0.465	0.825	0.946	69.1%	**0.951**	9.3%
2	0.069	0.259	0.359	0.828	73.2%	**0.903**	43.6%
3	0.961	0.488	0.953	**1.000**	99.7%	**1.000**	0.0%
4	0.903	0.489	0.853	0.956	54.6%	**0.971**	34.1%
$\overline{\mathbf{X}}$	0.685	0.425	0.747	0.933	74.2%	**0.956**	21.8%
Accuracy of non perturbated cells							
1	0.970	0.558	0.387	**0.990**	66.7%	**0.990**	0.0%
2	0.987	0.529	0.431	0.989	15.4%	**0.996**	63.6%
3	0.950	0.258	0.736	0.994	88.0%	**1.000**	100.0%
4	0.966	0.559	0.391	0.997	91.2%	**0.998**	33.3%
$\overline{\mathbf{X}}$	0.968	0.476	0.486	0.993	65.3%	**0.996**	49.2%
F-score							
1	0.882	0.507	0.527	0.968	72.9%	**0.970**	6.3%
2	0.128	0.348	0.392	0.902	83.9%	**0.947**	45.9%
3	0.955	0.337	0.831	0.997	93.3%	**1.000**	100.0%
4	0.934	0.522	0.536	0.976	63.6%	**0.984**	33.3%
$\overline{\mathbf{X}}$	0.725	0.429	0.571	0.961	78.4%	**0.975**	46.4%

Table 3.5 Refinement results for real data

	Individual			Ensemble			
QM	**minRSS**	**maxDiff**	**ALSC**	**DT**	**Gain%**	**BDT**	**Gain%**
Accuracy of perturbated cells							
1	0.485	0.167	0.515	**0.758**	50.1%	**0.758**	0.0%
2	0.345	0.093	0.564	0.618	12.4%	**0.764**	38.2%
3	0.212	0.091	0.364	**0.818**	71.4%	**0.818**	0.0%
4	0.394	0.111	0.576	0.576	0.0%	**0.818**	57.1%
$\overline{\mathbf{X}}$	0.359	0.115	0.505	0.692	33.5%	**0.789**	23.8%
Accuracy of non perturbated cells							
1	0.435	**0.670**	0.418	0.606	-19.4%	0.606	0.0%
2	0.875	0.929	0.110	0.956	38.0%	**0.966**	22.7%
3	0.661	**0.830**	0.219	0.785	-26.5%	0.752	-15.3%
4	0.520	**0.889**	0.148	0.546	-309.0%	0.658	24.7%
$\overline{\mathbf{X}}$	0.623	**0.829**	0.224	0.723	-79.2%	0.746	8.0%
F-score							
1	0.459	0.267	0.461	**0.673**	39.3%	**0.673**	0.0%
2	0.495	0.168	0.184	0.751	50.7%	**0.853**	41.0%
3	0.321	0.164	0.273	**0.801**	70.7%	0.784	-8.5%
4	0.448	0.198	0.235	0.560	20.3%	**0.730**	38.6%
$\overline{\mathbf{X}}$	0.431	0.199	0.288	0.696	45.3%	**0.760**	17.8%

And the error reduction gains of accuracy are calculated similarly. Compared with the score of the three individual refinement algorithms, minRSS, maxDiff, and ALSC, the combined ensemble learning of decision tree is very effective.

However, we find strong differences between the Q-matrices. For example, QM2 benefits of improvements close to 50% (QM2), while QM1 has a null improvement for real data and only 6.3% for synthetic data. In that respect, the boosting does not provide a gain that is as systematic as the one obtained from the DT which is positive for all matrices.

An important advantage of the boosting approach outlined here is that it can be applied to any classifiers. The base decision tree can also combine many algorithms to validate Q-matrices. Future work could look into combining more than the three algorithms of this study, and add new algorithms that potentially outperform them. And if the current results generalize, we would expect to make supplementary gains over any of them.

Moreover, the Q-matrices used in this research are quite small in size. The performance of boosted decision tree on larger Q-matrix and larger dataset would also be of interest.

Another interesting contribution is the demonstration that we can use ensemble algorithms such as the decision tree and Boosting by training over simulated data of Q-matrices. Without this synthetic data, training and extraction of features, such as the *stickiness* property, would not be feasible.

CHAPTER 4 Q-MATRIX DESIGN

4.1 Introduction

Regardless if student skill assessment is for classroom assessment or for online tutoring system, there is a strong incentive to minimize the number of questions. The number of questions needs to be large enough to correctly distinguish different student profiles, yet it also needs to be small enough to assess student profiles efficiently, avoiding too much work on both student side and tutors side. To tackle this problem, we see that there are two steps involved. First, the Q-matrix design needs to guarantee the feasibility of distinguishing different skill mastery levels. Since student profiles are parameters in statistical models, this question leads to the *identifiability* problem. Second, when identifiability requirement is satisfied, how does one find the best Q-matrix from a pool of candidates? More specifically, this problem can be rephrased as, given a set of skills to assess and a fixed number of question items, determine the optimal set of items, out of a potentially large pool, that will yield the most accurate assessment based on some criterion or loss function. Since all candidate Q-matrices are binary, this is a *discrete optimization* problem.

In recent years, the Q-matrix identifiability under DINA/DINO models has been proposed as a guiding principle for that purpose. We empirically investigate the extent to which identifiability can serve that purpose. Identifiability of Q-matrices is studied throughout a range of conditions in an effort to measure and understand its relation to student skills assessment. The investigation relies on simulation studies of skills assessment with synthetic data. Results show that identifiability is an important factor that determines the capacity of a Q-matrix to lead to accurate skills assessment with the least number of questions.

The rest of this chapter is organized as follows. In Section 4.2, a deterministic case with no uncertainty involved is analyzed for inspiration. In Section 4.3, the theoretical results of identifiabilities of DINA model parameters are given. In Section 4.4 and 4.5, two experiment results (P. Xu & Desmarais, 2018) are given to show the best Q-matrix design strategy. The conclusion is, the best Q-matrix design is to use only the unit vectors $\{e_i : i = 1, ..., K\}$ since it offers quicker convergence speed (as shown in experiment 1) and better robustness against slip and guess (as shown both in experiments 1 and 2). This conclusion is further solidified theoretically in Section 4.6.

4.2 Deterministic Case

Before talking about identifiability in a statistical framework, we first examine the deterministic case where no probability is involved thus identifiability becomes uniqueness in this case. For example, if slip=guess=0, then DINA model becomes a deterministic model. That is, if a student mastered all required skills for an item, then the answer will always be correct. We know that for this case, we have the binary decomposition formula

$$\boldsymbol{X} = \overline{\overline{\boldsymbol{P}} \odot \boldsymbol{Q}^{\boldsymbol{T}}}$$

where $\odot$ is denotes Boolean Matrix Product, the overline operator is the negation ($\overline{\boldsymbol{A}} = \boldsymbol{1} - \boldsymbol{A}$) and $\boldsymbol{P}$, $\boldsymbol{Q}$ denote the profile matrix and Q-matrix respectively.

Mathematically, finding a decomposition of a matrix $\boldsymbol{X}$ satisfying above formula is basically find a decomposition in the form below

$$\boldsymbol{Y} = \boldsymbol{A} \odot \boldsymbol{B}$$

where $\boldsymbol{Y} = \overline{\boldsymbol{X}}$, $\boldsymbol{A} = \overline{\boldsymbol{P}}$ and $\boldsymbol{B} = \boldsymbol{Q}^{\boldsymbol{T}}$. The question is, for a given result matrix $\boldsymbol{Y}$, how to find such a decomposition and is this decomposition unique?

This is the *Binary Matrix Factorization* (BMF) problem. Here we show that the decomposition is not unique by showing its equivalence to block covering problem. A **block** of a binary matrix is a submatrix where every entry is 1. For example, consider matrix $\boldsymbol{Y} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$, then $\boldsymbol{K}_1 = \{(1,3),(1,2)\} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ is a block. Here $\{(1,3),(1,2)\}$ means choosing the rows (1,3) and the columns (1,2). A block is said to **cover** its entries. A *block covering* of a binary matrix $\boldsymbol{Y}$ is a collection of blocks that any entry of $\boldsymbol{Y}$ is covered by at least one of the blocks. Then to find a Boolean matrix factorization of $\boldsymbol{Y}$ is equivalent to find a block covering of $\boldsymbol{Y}$ (Phelps, 1996). To show this, we can look at an example.

Consider the decomposition below of matrix $\boldsymbol{Y}$

$$\begin{aligned}
\boldsymbol{Y} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}
\end{aligned}$$

Here the Boolean matrix product symbol $\odot$ is omitted for simplicity. We can easily see from the above process that the decomposition is not unique. In fact we also have a different decomposition below

$$\begin{aligned}
\boldsymbol{Y} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}
\end{aligned}$$

We know that in the right side of this decomposition, the first matrix is correspondent to the profile matrix $\overline{\boldsymbol{P}}$ while the second matrix is correspondent to the Q-matrix $\boldsymbol{Q}^T$. We see thereby that this Q-matrix can not assess student profiles correctly since it can not give unique results.

Therefore, when we design a Q-matrix, we first need to guarantee that this Q-matrix can assess skills correctly, with no ambiguity. The above example is just a counterexample for genuine deterministic "DINA" , then what about the probabilistic DINA model? Fortunately, the theoretical results of identifiability of DINA model have already been established, which we will give in the next section.

4.3 Identifiability

First we need to define the notion of *complete* and *identifiable*.

Definition (Chiu et al., 2009; Chen, Liu, Xu, & Ying, 2015) A matrix Q is *complete* if $\{e_i : i = 1, ..., K\} \subset R_Q$, where R_Q is the set of row vectors of Q and e_i is a row vector such that the i-th element is one and the rest are zero (i.e. a binary unit vector).

Definition (Casella & Berger, 2002) A parameter θ for a family of distribution $f(x|\theta : \theta \in \Theta)$ is *identifiable* if distinct values of θ correspond to distinct pdfs or pmfs. That is, if $\theta \neq \theta'$, then $f(x|\theta)$ is not the same function of x as $f(x|\theta')$.

For the DINA model, the model parameters are $\mathbf{s}$, $\mathbf{g}$, $\mathbf{p}$ and $\boldsymbol{Q}$. Chen et al. (2015) established the identifibility results for all these parameters. The conditions can be listed below (Chen et al., 2015)

A1. $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_N$ are independently and identically distributed random vectors following distribution $p(\boldsymbol{\alpha}_i = \boldsymbol{\alpha}) = p_{\boldsymbol{\alpha}}$ and the population is fully diversified meaning that $p_{\boldsymbol{\alpha}} > 0$ for all $\boldsymbol{\alpha}$.

A2. All items have discriminating power meaning that $1 - s_j > g_j$ for all j.

A3. The true matrix $\boldsymbol{Q}_0$ is complete.

A4. Each attribute is required by at least two items, that is, $\sum_{j=1}^{J} q_{jk} \geq 2$ for all k.

A5. Each attribute of Q-matrix is associated to at least three items, that is, $\sum_{j=1}^{J} q_j k \geq 3$ for all k.

A6. $\boldsymbol{Q}$ has two complete submatrices, that is, for each attribute, there exists at least items requiring only that attribute. If so, the matrix can be rearranged into the form below

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{I_k} \\ \boldsymbol{I_k} \\ \boldsymbol{Q_1} \end{pmatrix}$$

The main theoretical results are

Theorem 1 (Liu, Xu, & Ying, 2013) For the DINA model, if the guessing parameters g_j's are known, under Conditions A1, A2 and A3, the Q-matrix is identifiable.

Theorem 2 (Chen et al., 2015) Under the setting of Theorem 1, the slipping parameters s_j and the attribute parameter $\boldsymbol{p}$ are identifiable if and only if Condition A4 holds.

Theorem 3 (Chen et al., 2015) Under the DINA and DINO models, with ($\boldsymbol{s}$, $\boldsymbol{g}$, $\boldsymbol{p}$) unknown, if Conditions A1,2,5 and 6 holds, then $\boldsymbol{Q}$, $\boldsymbol{s}$, $\boldsymbol{g}$, $\boldsymbol{p}$ are all identifiable.

4.4 Experiment 1: Comparison of three Q-matrix design strategies

From the discussion above we can see that identity matrix plays an import role in guaranteeing the identifiability of model parameters. In fact, when the uncertainty increases, we need more unit vectors in Q-matrix (Theorem 3). This observation inspires the idea that the Q-matrix design should also incorporate the identity matrix to guarantee the identifiability of student profile parameters, and the hypothesis is more unit vectors in Q-matrix can decrease the uncertainty in student profile diagnosis. This corresponds to a Q-matrix design strategy of repeating identity matrix.

To see how this strategy works, we compare it with two other strategies in both 3-skill and 4-skill cases, and we specify all three strategies below

- Strategy 1: Using the identifiability condition by only repeatedly using the vectors $\{\boldsymbol{e}_i : i = 1, ..., K\}$ (binary unit vectors, or one-hot encodings). Q-matrix used in this strategy is denoted as Q-matrix 1.
- Strategy 2: Using the vectors $\{\boldsymbol{e}_i : i = 1, ..., K\}$ plus an all-one vector $(1, 1, 1)$ (in the 3-skill case) or $(1, 1, 1, 1)$ (in 4-skill case). This is inspired by orthogonal array design, which is a commonly seen design of experiments (Montgomery, 2017). Q-matrix used in this strategy is denoted as Q-matrix 2.
- Strategy 3: Repeatedly using all q-vectors. Q-matrix used in this strategy is denoted as Q-matrix 3.

For the 3-skill case, all these three Q-matrices are shown in Figure 4.1. The general pattern is to recycle the rows above the lines denoted by ...[..., ..., ...]. The 4-skill case is similar, which is omitted here.

After defining the three strategies, we need to find a metric to compare their performance for assessing student profiles. We need to compare the profiles assessed from a DINA model based on the given Q-matrices with the true profiles. For the 3-skill case, there are 8 different

student profiles. We use one-hot encoding to denote all profile categories. Set M to be the number of profile categories. Then, in the 3-skill case, the $M = 8$ profile categories pc_i are:

$$\begin{array}{c} \\ pc_1 \\ pc_2 \\ pc_3 \\ pc_4 \\ pc_5 \\ pc_6 \\ pc_7 \\ pc_8 \end{array} \begin{array}{c} \begin{array}{ccc} k_1 & k_2 & k_3 \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{array}\right] \end{array}$$

Therefore, a student belonging to profile pc_1 is encoded as a binary unit vector $\alpha_1 = (1, 0, 0, 0, 0, 0, 0, 0)$, and so on for pc_2 encoded $\alpha_2 = (0, 1, 0, 0, 0, 0, 0, 0)$, ..., and pc_8 encoded $\alpha_8 = (0, 0, 0, 0, 0, 0, 0, 1)$. For this case, the DINA model parameter $\boldsymbol{p} = (p_1, p_2, ..., p_8) = (P(\alpha_1), P(\alpha_2), ..., P(\alpha_8))$ where P is a probability.

For a given profile α_i, for the DINA model, we have the likelihood

$$P(X_i|\alpha_i) = \prod_{j=1}^{J} P_j(\alpha_i)^{X_{ij}}[1 - P_j(\alpha_i)]^{1-X_{ij}}$$

and we need to conduct inference on α_i by maximizing

$$P(\alpha_i|X_i) \propto P(X_i|\alpha_i)P(\alpha_i)$$

This is essentially a MAP (Maximum a posteriori) estimation. Here we only have 8 possible α_i, and the prior of them is uniformly distributed, thus we can calculate the likelihood of each of them and choose the correspondent α_i yielding the highest likelihood as the MAP estimation of α_i. Now, for any Q-matrix configuration, the loss function is defined by

$$Loss(Q) = \sum_{i \in \text{students}} ||\hat{\alpha}_i - \alpha_{\text{true}}||_2$$

where $|| \cdot ||_2$ is the Euclidean norm.

Now the question that remains is how to generate the synthetic data? Since we are only concerned with DINA model, the dataset will also be generated under DINA model constraints. To generate student response data based on DINA model, we need to offer slip, guess, Q-matrix and student profiles. Here the student profiles are fixed, we use $N = 200$ student for 3-skill case, with each type of the 8 profile categories represented by 25 students. For the

4-skill case, we use $N = 400$, with each of type of the 16 profile categories also represented by 25 students. These profiles are the ground truth and we will estimate them by using DINA model again. The performance of different Q-matrix design strategies will be shown under different setting of $\boldsymbol{s}$ and $\boldsymbol{g}$.

We show the result of loss versus number of questions under different strategies, 3-skill and 4-skill cases in in Figure 4.2 and Figure 4.3 respectively.

From the result of this experiment we can see that strategy 1 always works better than the other two strategies, meaning that simply repeating the vectors $\{e_i : i = 1, ..., K\}$ in Q-matrix design, without using any combination of skills, yields better student diagnosis performance, which validates our hypothesis. However, this is a comparison among three particular Q-matrix design strategies. For a given pool of q-vectors, there are tons of different Q-matrix available to be used, can we have a more refined comparison on all of them? To answer this question, we conduct the second experiment below.

4.5 Experiment 2: Find best configuration

The second experiment takes the brute force approach. We directly examine all possible Q-matrix configurations. First, for a given pool of q-vectors to choose from and an integer indicating the number of questions, we need to know the number of possible configurations of Q-matrices we have. This is equivalent to a classical combinatorial problem, that is, to allocate distinguished balls (q-vectors) to indistinguished cells (questions). It can be easily computed by combinatorial coefficients and interpreted by using stars and bars methods. For example, in 3-skills case, we have 7 q-vectors, and if we have 4 questions to allocate them, then we have $\binom{4+7-1}{7-1} = 210$ possible configurations. This number grows up sharply as a number of questions increases or number of patterns increases. As a comparison, in the 4-skills case, if we have 5 questions to allocate them, then we have $\binom{5+15-1}{15-1} = 11628$ possible configurations.

The setting of experiment 2 is the same as experiment 1. For each configuration, we calculate the MAP estimation for all categories of each student, and compare with the one-hot encoding for their true categories. The total loss is reported as the performance index.

Since now the aim is to show a comparison among all Q-matrix design strategies, we show the results of 6 combinations of different numbers of skills and questions:

- 3-skills case, 4 questions: Figure 4.4, Figure 4.5
- 3-skills case, 8 questions: Figure 4.6, Figure 4.7

Q-matrix 1
(binary unit vectors)

	k_1	k_2	k_3
q_1	1	0	0
q_2	0	1	0
q_3	0	0	1
...	...	...	...
q_{19}	1	0	0
q_{20}	0	1	0
q_{21}	0	0	1

Q-matrix 2
(binary unit + all-1s vectors)

	k_1	k_2	k_3
q_1	1	0	0
q_2	0	1	0
q_3	0	0	1
q_4	1	1	1
...	...	...	...
q_{17}	1	0	0
q_{18}	0	1	0
q_{19}	0	0	1
q_{20}	1	1	1
q_{21}	1	0	0

Q-matrix 3
(all combinations)

	k_1	k_2	k_3
q_1	1	0	0
q_2	0	1	0
q_3	0	0	1
q_4	1	1	0
q_5	1	0	1
q_6	0	1	1
q_7	1	1	1
...	...	...	...
q_{15}	1	0	0
q_{16}	0	1	0
q_{17}	0	0	1
q_{18}	1	1	0
q_{19}	1	0	1
q_{20}	0	1	1
q_{21}	1	1	1

Figure 4.1 Q-matrix design strategies

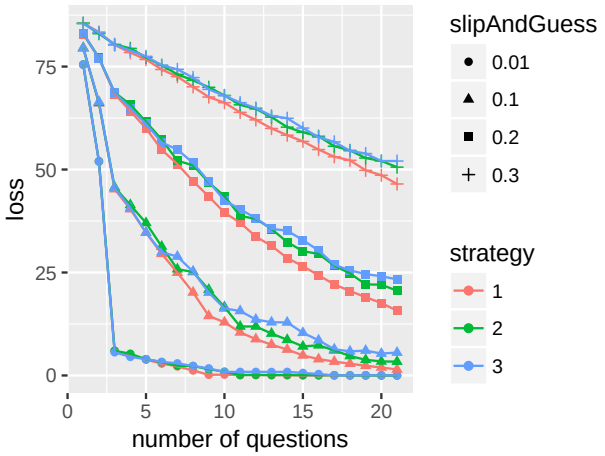


Figure 4.2 Three Strategy Comparison on 3-skill case

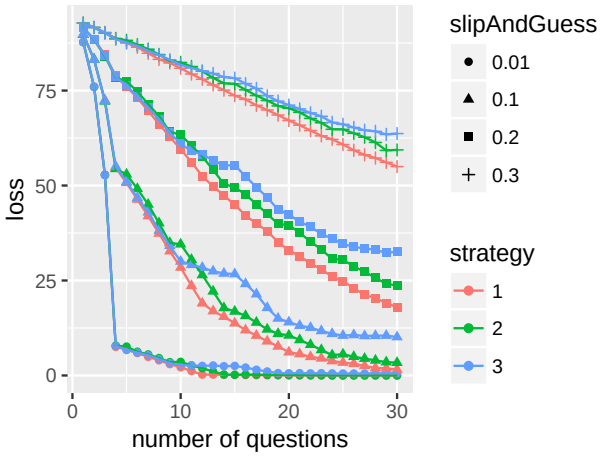


Figure 4.3 Three Strategy Comparison on 4-skill case

- 4-skills case, 5 questions: Figure 4.8, Figure 4.9

From the result of experiment 2, when slip and guess parameters are as low as 0.01, we can see obvious graded patterns among different configurations. This can be explained by the distinguishability of Q-matrix. For example, in Figure 4.4, we can see there are 7 layers. In fact, the first layer consisted of Q-matrix that can only cluster students into 2 categories. One example of such kind Q-matrix is:

$$\begin{array}{c} \\ q_1 \\ q_1 \\ q_1 \\ q_1 \end{array} \begin{array}{c} \begin{array}{ccc} k_1 & k_2 & k_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

This Q-matrix can only discriminate between a student that mastered skill 1 or not. We know that there are in fact 8 categories of students, the 7 layers in Figure 4.4 from top to bottom correspond to the Q-matrix that can separate students into 2 to 8 categories. We can see that complete Q-matrices always fall in the bottom layer, which concurs with our hypothesis. The 4-skills case is similar in Figure 4.8.

When slip and guess parameter increase, it makes the points becomes more divergent which can be easily seen by comparison between Figure 4.4 and Figure 4.5. In order to see some more details, we categorize three types of Q-matrices.

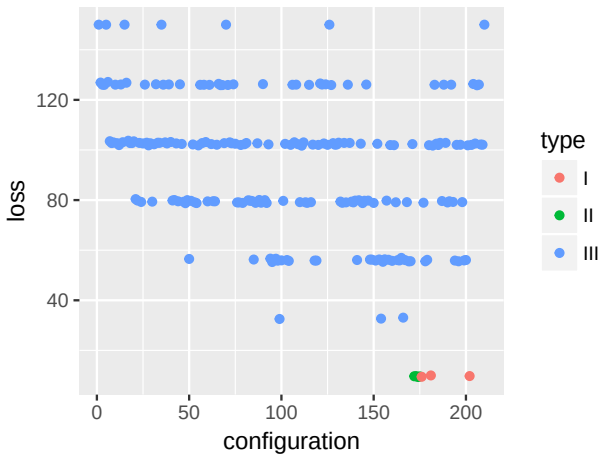


Figure 4.4 3-skill case, slip=guess=0.01, J=4

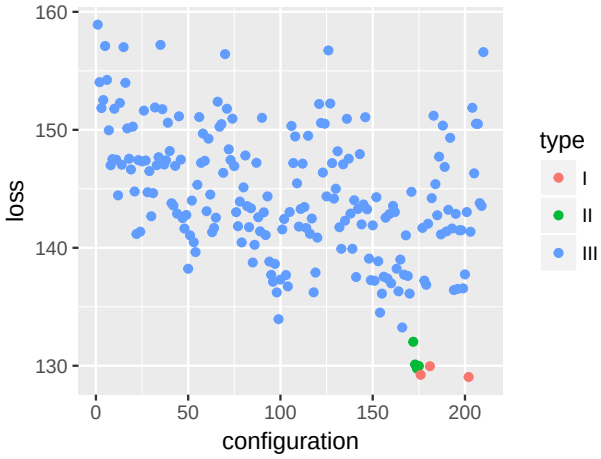


Figure 4.5 3-skill case, slip=guess=0.2, J=4

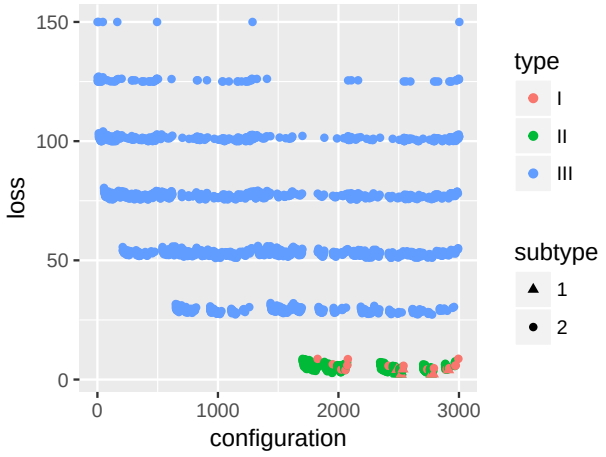


Figure 4.6 3-skill case, slip=guess=0.01, J=8

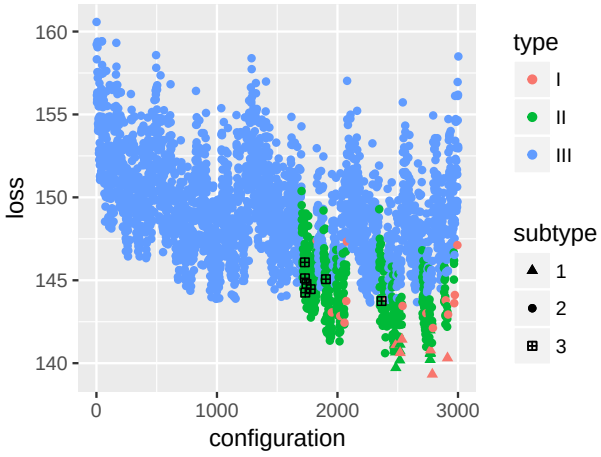


Figure 4.7 3-skill case, slip=guess=0.3, J=8

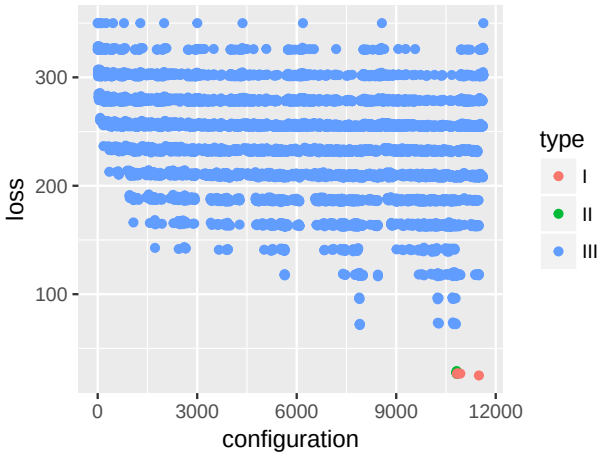


Figure 4.8 4-skill case, slip=guess=0.01, J=5

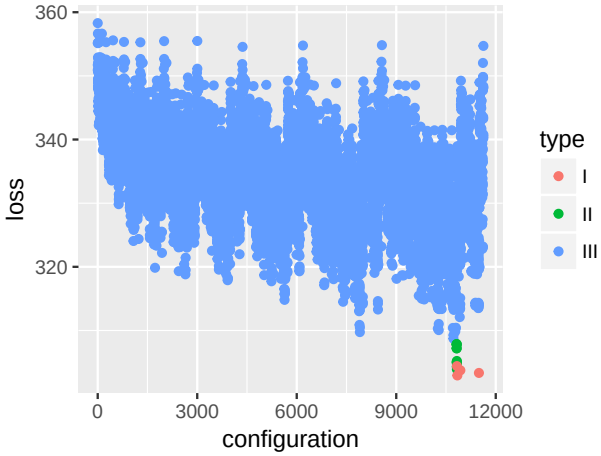


Figure 4.9 4-skill case, slip=guess=0.2, J=5

- Type I: Complete and confined, meaning it is only consisted of vectors $\{e_i : i = 1, ..., K\}$.
- Type II: Complete but not confined, meaning it not only contains all vectors $\{e_i : i = 1, ..., K\}$, but also contains at least one other q-vector.
- Type III: Incomplete Q-matrix.

Type I and Type II Q-matrices performs the same when slip and guess are low (Figure 4.4, Figure 4.8), but when they get higher, Type I Q-matrices show a better performance (Figure 4.5, Figure 4.9).

However, when more questions are involved in high slip and guess, the performance becomes more unstable. Therefore, we again consider more subtypes. In 3-skills case for 8 questions, we consider three subtypes below.

- Subtype 1: Q-matrix contains each component of $\{e_i : i = 1, ..., K\}$ at least twice.
- Subtype 2: Other situations (e.g a complete Q-matrix but all the other vectors are just repeated e_1).
- Subtype 3: Q-matrix contains all q-vectors.

From Figure 4.7 we can see that the subtype 1 (denoted by triangle) shows better performance than subtype 2, meaning that repeating the whole set of $\{e_i : i = 1, ..., K\}$ is a better strategy just like the strategy 1 we used in experiment 1. Subtype 3 corresponds to the strategy 3 in

experiment 1, it has only 7 possible configurations in 8-question setting and we can see that they do not perform well.

Therefore, we argue that the best Q-matrix design is to use only the vectors $\{e_i : i = 1, ..., K\}$ since it offers quicker convergence speed (as shown in experiment 1) and better robustness against slip and guess (as shown both in experiments 1 and 2).

4.6 Theoretical Discussion

After seeing the promising result of the two experiments, we would like to offer a theoretical explanation to the results. For simplicity, here we consider the 3-skill case. The profile of a student can be written as $(\alpha_1, \alpha_2, \alpha_3)$, in which α_i is a Bernoulli variable indicating whether the student has mastered skill i or not. Let $p(\alpha_i = 1) = \theta_i, i = 1, 2, 3$, then $\theta_i, i = 1, 2, 3$ are the parameters we want to learn. Without loss of generality, we choose a focus attribute α_1, and we want to know the effect of different choices of questions on the uncertainty of α_1. Here we use entropy $H(\alpha_1)$ to measure the uncertainty, that is $H(\alpha_1) = -\sum_{\alpha_1} p(\alpha_1) \log p(\alpha_1)$. Now with different questions answered, we will have different entropy of α_1, in other words, different conditional entropy. Below we prove that $H(\alpha_1|r_1 = 1) < H(\alpha_1|r_{12} = 1)$ which indicates that answering question $q_1 = (1, 0, 0)$ correctly will give less uncertainty on α_1 than answering question $q_{12} = (1, 1, 0)$ correctly.

First consider question $q_1 = (1, 0, 0)$, the response to this question is denoted as r_1, we have

$$\begin{aligned} p(\alpha_1 = 1|r_1 = 1) &= \frac{p(r_1 = 1|\alpha_1 = 1)p(\alpha_1 = 1)}{p(r_1 = 1|\alpha_1 = 1)p(\alpha_1 = 1) + p(r_1 = 1|\alpha_1 = 0)p(\alpha_1 = 0)} \\ &= \frac{(1 - s)\theta_1}{(1 - s)\theta_1 + g(1 - \theta_1)} \end{aligned} \tag{4.1}$$

And we have

$$\begin{aligned} p(\alpha_1 = 0|r_1 = 1) &= 1 - p(\alpha_1 = 1|r_1 = 1) \\ &= \frac{g(1 - \theta_1)}{(1 - s)\theta_1 + g(1 - \theta_1)} \end{aligned} \tag{4.2}$$

$$\begin{aligned} p(\alpha_1 = 1|r_1 = 0) &= \frac{p(r_1 = 0|\alpha_1 = 1)p(\alpha_1 = 1)}{p(r_1 = 0|\alpha_1 = 1)p(\alpha_1 = 1) + p(r_1 = 0|\alpha_1 = 0)p(\alpha_1 = 0)} \\ &= \frac{s\theta_1}{s\theta_1 + (1 - g)(1 - \theta_1)} \end{aligned} \tag{4.3}$$

$$\begin{aligned} p(\alpha_1 = 0|r_1 = 0) &= 1 - p(\alpha_1 = 1|r_1 = 0) \\ &= \frac{(1-g)(1-\theta_1)}{s\theta_1 + (1-g)(1-\theta_1)} \end{aligned} \tag{4.4}$$

Then consider question $q_{12} = (1, 1, 0)$, the response to this question is denoted as r_{12}, we have

$$p(\alpha_1 = 1|r_{12} = 1) = \frac{p(r_{12} = 1|\alpha_1 = 1)p(\alpha_1 = 1)}{p(r_{12} = 1|\alpha_1 = 1)p(\alpha_1 = 1) + p(r_{12} = 1|\alpha_1 = 0)p(\alpha_1 = 0)} \tag{4.5}$$

in which

$$\begin{aligned} p(r_{12} = 1|\alpha_1 = 1) &= p(r_{12} = 1|\alpha_1 = 1, \alpha_2 = 1)p(\alpha_2 = 1) + p(r_{12} = 1|\alpha_1 = 1, \alpha_2 = 0)p(\alpha_2 = 0) \\ &= (1-s)\theta_2 + g(1-\theta_2) \end{aligned} \tag{4.6}$$

and

$$\begin{aligned} p(r_{12} = 1|\alpha_1 = 0) &= p(r_{12} = 1|\alpha_1 = 0, \alpha_2 = 1)p(\alpha_2 = 1) + p(r_{12} = 1|\alpha_1 = 0, \alpha_2 = 0)p(\alpha_2 = 0) \\ &= g\theta_2 + g(1-\theta_2) \\ &= g \end{aligned} \tag{4.7}$$

Therefore

$$p(\alpha_1 = 1|r_{12} = 1) = \frac{((1-s)\theta_2 + g(1-\theta_2))\theta_1}{((1-s)\theta_2 + g(1-\theta_2))\theta_1 + g(1-\theta_1)} \tag{4.8}$$

We want to compare $H(\alpha_1|r_1 = 1)$ with $H(\alpha_1|r_{12} = 1)$. Notice that α_1 is a Bernoulli variable, we can just compare $p(\alpha_1 = 1|r_1 = 1)$ with $p(\alpha_1 = 1|r_{12} = 1)$ instead. In fact, from the graph of entropy function of Bernoulli variable we know that when $p < 0.5$, $H(X)$ is monotonically increasing while when $p > 0.5$, $H(X)$ is monotonically decreasing.

Now let us look at the two expressions (4.1) and (4.8), they share the same structure, in fact they can be seen as the values of function $f(T) = \frac{T\theta_1}{T\theta_1 + g(1-\theta_1)}$ at point $1 - s$ and $(1-s)\theta_2 + g(1-\theta_2)$ respectively. We can rewrite $f(T)$ as $f(T) = \frac{T}{T+k} = 1 - \frac{1}{T+k}$ where $k = \frac{g(1-\theta_1)}{\theta_1}$, then we can easily see this is a hyperbola with asymptote at $x = -k = -\frac{g(1-\theta_1)}{\theta_1} < 0$ and the function increases in both two branches.

Now we only need to compare $1-s$ with $(1-s)\theta_2 + g(1-\theta_2)$ which can be easily figured out since the latter is a convex combination of point $1-s$ and g, thus its value is between g and $1-s$, supposed we have $0 < s, g < 0.5$.

Now we only need to show that both $p(\alpha_1 = 1|r_1 = 1)$ and $p(\alpha_1 = 1|r_{12} = 1)$ are bigger than 0.5 so their values can be easily compared in $H(\alpha_1)$. In fact, we have

$$p(\alpha_1 = 1|r_1 = 1) = \frac{1-s}{(1-s)\theta_1 + g(1-\theta_1)}\theta_1 \tag{4.9}$$

We see that $p(\alpha_1 = 1|r_1 = 1) > \theta_1$ since $\frac{1-s}{(1-s)\theta_1+g(1-\theta_1)} > 1$ considered that the denominator is a convex combination of point $1-s$ and g. The same reasoning holds for $p(\alpha_1|r_{12} = 1) > \theta_1$ too. If the previous knowledge of θ_1 or the prior of θ_1 is $\theta_1 = 0.5$ then we have both $p(\alpha_1 = 1|r_1 = 1) > 0.5$ and $p(\alpha_1 = 1|r_{12} = 1) > 0.5$.

Now, from all the discussion above, we conclude that $H(\alpha_1|r_1 = 1) < H(\alpha_1|r_{12} = 1)$, which means tested by item $q_1 = (1, 0, 0)$ will make the skill mastery state α_1 less uncertain than tested by item $q_{12} = (1, 1, 0)$. Therefore, it theoretically solidifies the strategy of using unit vectors in Q-matrix design.

CHAPTER 5 Q-MATRIX DERIVATION

5.1 Introduction

For educational assessment, the item-skill mapping matrix, namely Q-matrix, is often needed to determine the status of skill mastery of a student. Typically, this Q-matrix is given by experts. However, different experts might give different Q-matrices. Even for a single expert, one might not be so sure of the given Q-matrix. Therefore, can we directly obtain a Q-matrix from the students response data? In other words, can we empirically estimate the Q-matrix from the data itself, instead of relying on an artificial one?

In fact, experts can just add these Q-matrix derivation techniques into their tool box for initiating a reasonable Q-matrix, which helps to build their own Q-matrix.

One way to tackle the Q-matrix derivation problem is to convert it into a Q-matrix refinement problem which we discussed in Chapter 3. A natural thought is starting from a random Q-matrix, pretending it to be an expert-given matrix and then use Q-matrix refinement method to obtain a new one. However, the feasibility of those refinement methods under this situation is not well researched, and due to the identifiability problem, we may expect unsatisfactory results. Therefore, researchers have developed other methods for deriving Q-matrix directly from student performance data, and in this chapter we will discuss them, in addition to proposing our own new method.

We first review several Q-matrix derivation methods in previous research, including Hill Climbing in Section 5.2, LASSO in Section 5.3, NMF in Section 5.4 and ALS in Section 5.5. Then we propose our new method, which we name *ClusterToQ*. It consists of two major steps of novelty, the first one is to use clustering techniques to obtain a set of ideal response patterns, and then to use a proposed algorithm to convert those patterns into a Q-matrix. Both advantages and disadvantages of this method compared to other methods are discussed in Section 5.6.

5.2 Hill Climbing

If we think each Q-matrix is associated with a "cost", and our aim is to find a Q-matrix from all possible ones that minimize a predefined "cost", then the Q-matrix derivation problem becomes an optimization problem. And more specifically, a discrete optimization problem if we require the entry of the Q-matrix to be binary. Using terms of discrete optimization, each

Q-matrix is a *candidate solution*, and our aim is to find the one that minimizes its associated cost.

However, just like many discrete optimization problems, the size of the *search space*, or the *feasible region* is very big. For example, consider a Q-matrix with 9 items and 3 skills, since each of its entry can be 0 or 1, then there are 2^{27} possible Q-matrices (over 130M). Considering that each matrix must be evaluated, often through a cross validation process, the computations over all possible Q-matrices would thereby become impractical.

One way to solve this problem is using heuristics. *Hill Climbing*, as a method of *local search*, is a commonly used heuristic algorithm in mathematical optimization. Barnes (2010) applied this algorithm into Q-matrix derivation problem. The idea follows a typical local search scheme. First we start with a random Q-matrix and calculate the cost associated with it, then apply a small change on a random cell of the Q-matrix and calculate the new cost associated with this new Q-matrix, if the new candidate solution yields lower cost, then keep the update, otherwise abandon it. This process iterates until no further improvement can be made. A pseudo-code is given in Algorithm 2.

There are several concerns of this algorithm. First is the definition of *cost*, which is called *error* in (Barnes, 2010). To calculate the error of each Q-matrix configuration, the IRP method discussed in Section 2 is used, that is, associate each student response vector to the closest ideal response pattern, which is called IDR (Ideal Response Vector) in (Barnes, 2010). The distance used to measure the closeness is ℓ_1 distance, that is

$$d(RESP, IDR) = \sum_k |RESP(k) - IDR(k)|$$

where k is the index of skills, $RESP(k)$ is the k-th entry of the response vector, and $IDR(k)$ is the k-th entry of the IDR.

The second important problem is the choice of the small change *delta* involved in the search process. Since a Q-matrix is binary, the commonly used small real number change will sabotage this limitation. Therefore, Barnes (2010) relaxed the binary limitation by allowing Q-matrix entries to be real values ranging from 0 to 1. But doing this needs a new interpretation of a Q-matrix, and the author used one from (Brewer, 1996), which is, each Q-matrix entry indicates the probability that a student will miss a question component incorrectly if he or she does not master that skill. We can see this idea consider questions in a component level. From a DINA model point of view, this can also be parameterized as $1 - g$, but on the skill level, not the item level. For example, if $Q[3][1] = 0.3, Q[3][2] = 0.2, Q[3][3] = 1$, it means for a student who does not master any of the three skills, he or she will miss the

Algorithm 2: Hill Climbing for Q-matrix Derivation (Barnes, 2010)

```
1. Set MinError = LargeNumber;
2. for Starts ← 1 to NumStarts do
   (a) Random initialization of Q[J][K] where J is the number of questions and K is
    the number of skills;
   (b) Set Q* = Q, CurrError = Error(Q);
   (c) for Iters ← 1 to NumIters do
       for k ← 1 to K do
           for j ← 1 to J do
               Q*[j][k] = Q[j][k] + delta;
               if Error(Q*) < CurrError then
                   do
                       set Q* = Q;
                       set CurrError = Error(Q*);
                       Q*[j][k] = Q[j][k] + delta;
                   while Error(Q*) < CurrError;
               else
                   Q*[j][k] = Q[j][k] − delta ;
                   while Error(Q*) < CurrError do
                       set Q* = Q;
                       set CurrError = Error(Q*);
                       Q*[j][k] = Q[j][k] − delta;
                   end
               end
           end
       end
   end
   (d) if CurrError < MinError then
       set BestQ = Q;
       set minError = currError;
end
```

first component of question 3 with probability 0.3, the second component with probability 0.2 and third component with probability 1. Then the probability of student i to get item 3 correctly is $P(X_{i3} = 1) = (1 - 0.3) * (1 - 0.2) * (1 - 1) = 0$.

The third problem is how to avoid getting stuck in local optima, since it is an inherent feature of local search heuristics. Barnes (2010) used the traditional way, by trying multiple random starts, and choosing the best performing one.

The fourth problem is whether to fix the number of skills, since it is also a latent variable. The author did not fix it, but increase it gradually starting from 1. Then there is the problem of when to stop increasing. The author proposed to preset a threshold error, like less than 1 error for every student. The increase of number of skills stops when this criterion is reached.

This HC method is used for comparison in our later experiments, but we fixed the number of skills in advance since our main aim is to compare different Q-matrix derivation algorithms for the same datasets.

5.3 LASSO

Another way to tackle the Q-matrix derivation problem is to treat it as a variable selection problem in a statistical model (Chen et al., 2015). We survey this approach by first reviewing an alternative representation of DINA model given in Chen et al. (2015).

The item response function of DINA model can be alternatively written as:

$$\begin{aligned}\theta_{j,\boldsymbol{\alpha}} = P(X^j = 1|\boldsymbol{\alpha}, \beta^j) =& \text{logit}^{-1}(\beta_0^j + \sum_{k=1}^{K} \beta_k^j \alpha_k + \sum_{1 \le k_1 < k_2 \le K} \beta_{k_1 k_2}^j \alpha_{k_1} \alpha_{k_2} \\ &+ \sum_{1 \le k_1 < k_2 < k_3 \le K} \beta_{k_1 k_2 k_3}^j \alpha_{k_1} \alpha_{k_2} \alpha_{k_3} + ... + \beta_{12...K}^j \prod_{k=1}^{K} \alpha_k)\end{aligned} \tag{5.1}$$

where $\text{logit}(p) = \log \frac{p}{1-p}$ for $p \in (0, 1)$, and $\text{logit}^{-1}(x) = \text{logistic}(x) = \frac{1}{1+e^{-x}}$ where $x \in R$. Notice here the item index j is moved from subscript to superscript to be consistent with the original paper. More precisely, X_i^j is the response data of student i towards item j and here i is omitted for the specific profile pattern $\boldsymbol{\alpha}$. We can see this is a *Generalized Linear Model* (GLM) but with all interaction terms.

DINA model can be considered as a special case in this expression where only one coefficient is non-zero besides the constant term β_0^j. For example, if β_{23}^j is non-zero, then the q-vector for $j - th$ item has 1 for the 2nd and 3rd skills, and 0 for all the remaining skills. For this

item, the item response function is

$$\theta_{j,\boldsymbol{\alpha}} = \text{logit}^{-1}(\beta_0^j + \beta_{23}^j \alpha_2 \alpha_3)$$

Notice that in original DINA model expression, the item response function is:

$$\theta_{j,\boldsymbol{\alpha}} = P(X^j = 1|\boldsymbol{\alpha}, g_j, s_j) = \begin{cases} g_j, & \text{if at least one skill not mastered} \\ 1 - s_j, & \text{if all skill mastered} \end{cases} \tag{5.2}$$

For our example, when $\alpha_2 = 1$ and $\alpha_3 = 1$, then $\theta_{j,\alpha} = 1 - s_j = \frac{e^{\beta_0^j + \beta_{23}^j}}{1+e^{\beta_0^j + \beta_{23}^j}}$; otherwise $\theta_{j,\alpha} = g_j = \frac{e^{\beta_0^j}}{1+e^{\beta_0^j}}$. Thus we see how the parameters of DINA model, i.e. g_j and s_j are expressed by the new parameters β^j. In fact, the original parameters s_j and g_j in this example are replaced by parameters β_0^j and β_{23}^j.

From the discussion above, we see that the non-zeros of coefficients β^j determine the structure of q-vectors. In fact, for DINA model, only the constant β_0^j and one other coefficient should be non-zero. Therefore, if we can find a way to estimate those coefficients while forcing most of them to be 0, then we can obtain the correspondent Q-matrix. By doing this, we actually convert the Q-matrix derivation problem into a variable selection problem.

Fortunately, for generalized linear model, LASSO (*Least Absolute Shrinkage and Selection Operator*) is a popular approach to do so. Originally proposed in (Tibshirani, 1996), LASSO is a general way to do variable selection by regularizing likelihood through adding a ℓ_1 penalty term. The consistency of LASSO, that is, the condition of LASSO to find the true model by variable selection is developed in Fan and Li (2001); Zhao and Yu (2006); Fan and Lv (2011). Compared to ℓ_2 regularization, or ridge regression, the main property of LASSO is it can force the irrelevant parameters to be 0, thus achieving the aim of variable selection.

Therefore, a regularized MLE (*Maximum Likelihood Estimation*) of the parameters is given by:

$$(\hat{\beta}^1, ...\hat{\beta}^J) = \underset{\beta^1,...\beta^J}{\arg\max} \log[L(\beta^1, ...\beta^J; \boldsymbol{X}_i, i = 1, ..., N)] - N \sum_{j=1}^{J} p_{\lambda_j}(\beta^j) \tag{5.3}$$

where p_{λ_j} is the penalty function and λ_j is the regularization parameter. In our research, we use LASSO regularization, namely the second term:

$$p_\lambda(\beta) = \lambda \sum_{k=1}^{K} |\beta_k|$$

and since the conditional likelihood function is:

$$L(\beta^1, ...\beta^J; \boldsymbol{X}_i, \boldsymbol{\alpha_i}, i = 1, ..., N) = \prod_{i,j} (\theta_{j,\boldsymbol{\alpha_i}})^{X_i^j} (1 - \theta_{j,\boldsymbol{\alpha_i}})^{1-X_i^j}) \tag{5.4}$$

then the first term in (5.3), or the observed data likelihood, is:

$$L(\beta^1, ...\beta^J; \boldsymbol{X}_i, i = 1, ..., N) = \prod_{i,j}^{N} \sum_{\boldsymbol{\alpha}_i} [p_{\boldsymbol{\alpha}_i} (\theta_{j,\boldsymbol{\alpha}_i})^{X_i^j} (1 - \theta_{j,\boldsymbol{\alpha}_i})^{1-X_i^j})] \tag{5.5}$$

Since this optimization is again involved with latent variables, EM algorithm is used for calculation.

5.4 NMF

Initially proposed in Paatero and Tapper (1994); Paatero (1997) and flourished after (Lee & Seung, 1999, 2001), Nonnegative Matrix Factorization (NMF) is now a popular method used in many fields, including document clustering (W. Xu, Liu, & Gong, 2003; Shahnaz, Berry, Pauca, & Plemmons, 2006), image classification (Gupta & Xiao, 2011), stock market pricing (de Fréin, Drakakis, Rickard, & Cichocki, 2008), etc. Using NMF for Q-matrix derivation has also been explored in M. Desmarais (2011); M. C. Desmarais (2012). A more detailed overview of NMF and its applications can be found in Wang and Zhang (2013).

In its classic form, NMF tries to decompose a nonnegative matrix into a product of two nonnegative matrices. Because of its nonnegativity, the original matrix can be considered as a combination of parts for which only additive relation is allowed. This usually leads to a more meaningful interpretation than ordinary matrix factorization like those used in Principle Component Analysis (PCA) or Vector Quantization (VQ). For example in Lee and Seung (1999), the component matrix obtained by NMF can be interpreted as parts of an image, while the component matrix obtained by PCA or VQ are interpreted as wholistic features.

This additive constraint in NMF has a similarity in DINA model, or in general, conjunctive models. Therefore, we can interpret one component matrix as skill parts and the other as each student profile of these skills. If we binarize the skill parts component matrix, we again obtain the Q-matrix.

5.5 ALS

Just like the hill climbing method mentioned above that uses random starts as initial Q-matrices, we can apply this strategy to ALS method that was originally used for Q-matrix refinement too, which also serves as an option to derive Q-matrix. More specifically, we start with a random binary matrix instead of expert given Q-matrix, then minimize $||\boldsymbol{X} - \boldsymbol{P}\boldsymbol{Q}^{\boldsymbol{T}}||$ with respect to $\boldsymbol{P}$ and $\boldsymbol{Q}$ alternatively. The final result is binarized to obtain a meaningful Q-matrix. Details of ALS can be found in Section 3.4.

5.6 From Clustering to Q-matrix

Inspired by the nonparametric method used in student profile diagnosis (Chiu & Douglas, 2013) discussed in Section 2.4 and Q-matrix refinement (Chiu, 2013) discussed in Section 3.3, we propose to use a nonparametric method to find Q-matrix. The core idea is to associate Q-matrix with ideal response patterns, which is obtained by clustering the observed student performance data and extract information from the cluster centers. Details are given below.

5.6.1 Clustering

The first step is to obtain the unobserved ideal response patterns. This is done through clustering. And the center of each cluster is interpreted as an ideal response pattern.

Cluster analysis is not new to psychometrics and educational assessment. A typical application of it can be found in Chiu et al. (2009). In that research, each student response vector is transformed into a *vector of sum-scores*, which is the score of the student on each skill. This calculation is feasible because the Q-matrix is given. Two methods were considered for clustering, which are *k-means* and *Hierachical Agglomerative Cluster Analysis* (HACA). However, that research is dealing with the case where the Q-matrix is already given, thus it is different from our situation.

In our research, the observed data is solely the student performance data, and we want to cluster them to obtain the ideal response patterns. Since the quality of clustering is critical to the second step of our method, We compare multiple clustering techniques here. First, we consider model based clustering method, since the response data is binary, we use the Bernoulli distribution based model, which is called *Mixture of Multivariate Bernoulli* (MMB) model. Second we consider partition based clustering method by discussing the popular k-means method, and its variation, the k-medoids method.

MMB, Mixture of Multivariate Bernoulli

Models with hidden variables are called *Latent Variable Models* (LVM). Denote the latent variable for i-th sample as z_i, if the distribution of z_i is discrete, then this simple LVM is called a *Mixture Model* (McLachlan, 1988; Bishop, 1994, 2006), which is also called *Latent Class Model* in the statistics and psychometrics community (Roussos, Templin, & Henson, 2007; Chen, Li, Liu, & Ying, 2017; G. Xu & Shang, 2018; De Menezes, 1999). Generally, the conditional likelihood of a mixture model can be written as:

$$p(\boldsymbol{x}_i|z_i = k) = p_k(\boldsymbol{x}_i)$$

where p_k is the k'th base distribution and can be of any type. It is called a mixture model, because we are mixing K base distributions in its likelihood:

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x}_i|\boldsymbol{\theta})$$

We can require the base distribution of mixture model to be a product of Bernoulli, that is:

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{J} Ber(x_{ij}|\mu_{jk}) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{J} \mu_{jk}^{x_{ij}}(1-\mu_{jk})^{1-x_{ij}}$$

where μ_{jk} is the Bernoulli parameter, or the probability of the k-th cluster to get the j-th item to be 1, then this model is called *Mixture of Multivariate Bernoulli* (MMB) model (Murphy, 2012). The parameters of this model can be learned by the EM algorithm.

k-means and k-medoids

Originated from signal processing, k-means algorithm is commonly used for clustering. It has an intuitive implementation. First initialize k centers. Then assign each sample into its closest cluster. After that, the center of each cluster is updated by the mean value of its assigned samples. The whole process is repeated until convergence.

If the distance used to calculate the closeness in k-means clustering is Euclidean, then k-means algorithm also has a probabilistic interpretation. In fact, it is a simplified version of the EM algorithm. In the above mentioned mixture model, if we require each base distribution to be a Gaussian distribution, that is:

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where $\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the k'th Gaussian distribution, and we require $\boldsymbol{\Sigma}_k = \sigma^2 \boldsymbol{I}_J$ to be fixed, and $\pi_k = 1/K$ to be also fixed, then we only have parameters μ_k left to be learned. Using the EM algorithm to learn this model will yield the classic k-means algorithm (Murphy, 2012).

Ignoring the probabilistic interpretation, we will use two different ways to calculate distance in the k-means algorithm. One is the ℓ_2 distance or the Euclidean distance, the other is the ℓ_1 distance or the Manhattan distance.

If we always choose a representative sample as the center, instead of using the means, then we obtain the k-medoids algorithm (Park & Jun, 2009). So instead of calculating the mean, we calculate the "cost" of assigning one sample as the cluster center. This "cost" is a sum of distances between the chosen sample and all other in-cluster samples. Since it also involves calculating distance, we also test it with both ℓ_2 distance and ℓ_1 distance.

To compare the performance of different clustering techniques, we use four metrics to evaluate their results. In spite of the true cluster centers being binary, all clustering techniques except k-medoids yield real-value cluster centers. Therefore, we first compare them by RMSE, then after binarization, we compare them by F-score, cell-wise accuracy and vector-wise accuracy. The definition of F-score was given in Section 3.5, and the definition of other metrics are given below

$$\text{RMSE} = \sqrt{\sum_i \sum_j (IRP_{ij}^{estimated} - IRP_{ij}^{true})^2} \tag{5.6}$$

$$\text{Cell-wise Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}} \tag{5.7}$$

In cell-wise case, each correctly predicted entry 1 or 0 is considered as a true positive or true negative. But in vector-wise case, only when all entries in an IRP are predicted correctly would that IRP be considered as a correct prediction. We can see here that it is meaningless to distinguish true positives and true negatives in vector-wise case, thus we only need to consider the accuracy and ignore the F-score. To summarize, the vector-wise accuracy here is defined as

$$\text{Vector-wise Accuracy} = \frac{\text{Number of Correctly Predicted IRPs}}{\text{Number of All Predicted IRPs}} \tag{5.8}$$

Note that there is no order of those IRPs, thus we need to align IRPs before comparison. Here since we are dealing with synthetic data, we thereby have the true IRP matrix (each

row is an IRP). Consequently, we will align the IRPs obtained by clustering methods to the true IRP matrix. The alignment is done simply by comparing each IRP to the true IRP matrix and choose the closest IRP one by one.

Experiment

We run experiments for different configurations of parameters. First, we consider both 3-skill and 4-skill cases. In the 3-skill case, we consider two sample sizes for students, $N = 100$ and $N = 500$, while in 4-skill case, we consider the case $N = 200$ and $N = 500$. For each of them, we check the performance under different slip and guess values of 0.05, 0.1 and 0.2 respectively. To summarize, we have the following 12 configurations:

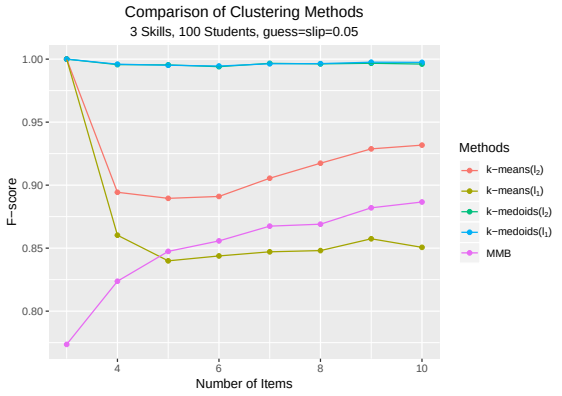
- 3-skill, students=$\{100,500\}$, slip=guess=$\{0.05, 0.1, 0.2\}$ (6 cases)
- 4-skill, students=$\{200,500\}$, slip=guess=$\{0.05, 0.1, 0.2\}$ (6 cases)

Inside each configuration, we consider different number of items. In 3-skill case, we consider numbers from 3 to 10, while in 4-skill case, we consider numbers of 4 to 20. For each number case of items, first we generate a random Q-matrix which satisfies the identifiability requirement. Then we generate the student profiles uniformly. In fact, for the 3-skill case, the student profiles are generated uniformly from 8 patterns, while in the 4-skill case, the student profiles are generated uniformly from 16 patterns. With the student profiles and Q-matrix we can generate the performance matrix given slip and guess parameters. Then we apply different clustering techniques on the performance matrix. When multiple random starts are needed like in k-means, 10 starts are used. This process is run 128 times for each number of items in each configuration, and the average performance is reported as final results. The detailed results are given in Appendix A. Here we show the figures of cell-wise F-score and vector-wise accuracy for a visual comparison.

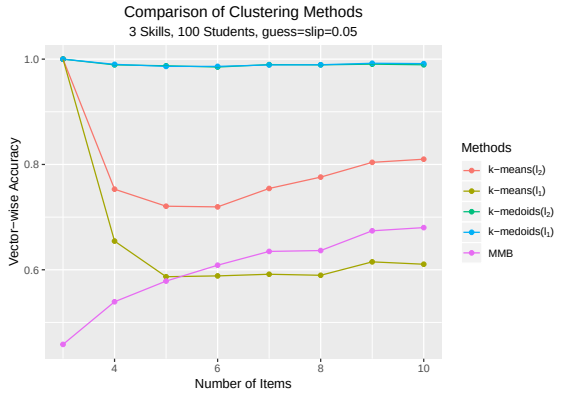
Results

Now let us look at the results of the step 1 of our method, that is, the results of clustering.

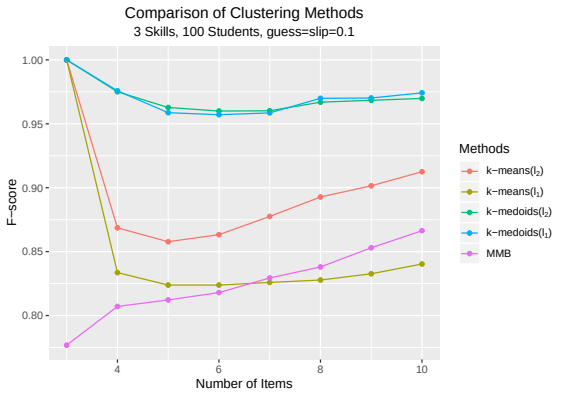
We can see that when slip and guess are as low as 0.05, then with more items tested, we get better performance with all clustering methods. But when slip and guess are high, at 0.2, we get worse performance with all methods with more items tested. This observation matches intuition, since the IRPs are clear when the performance data is of low randomness, but would be much disturbed when the randomness is high in performance data, thus making clustering methods difficult to find true IRPs.



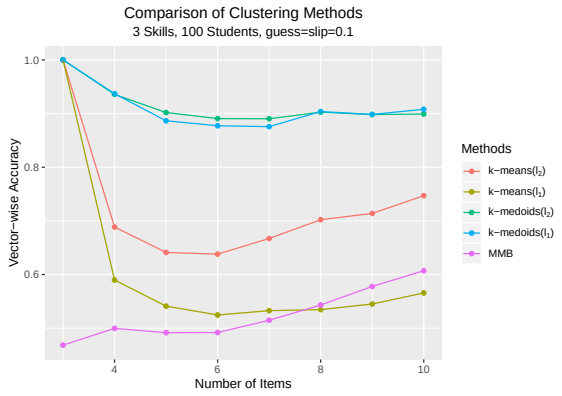
(a) F-score for cell-wise guess=slip=0.05



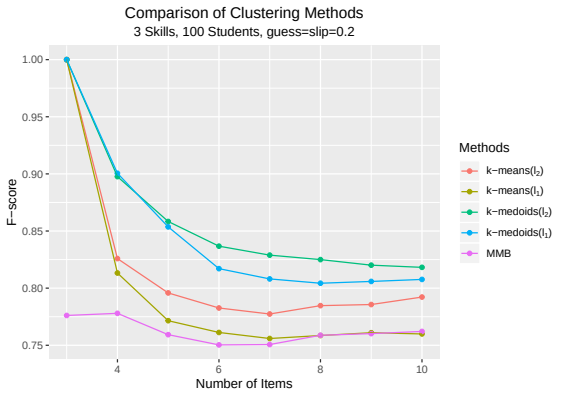
(b) Vector-wise Accuracy for guess=slip=0.05



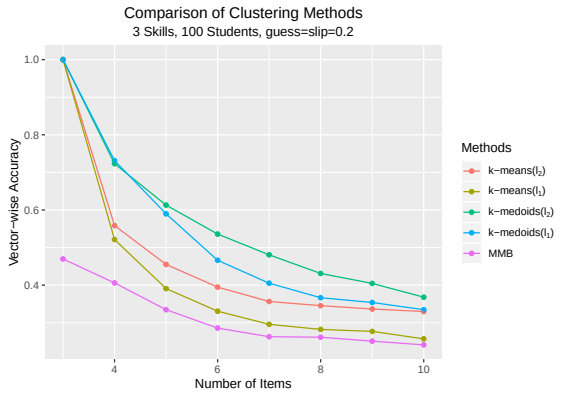
(c) F-score for cell-wise guess=slip=0.1



(d) Vector-wise Accuracy for guess=slip=0.1

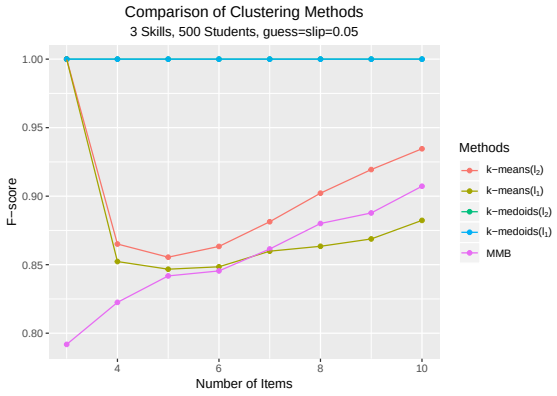


(e) F-score for cell-wise guess=slip=0.2

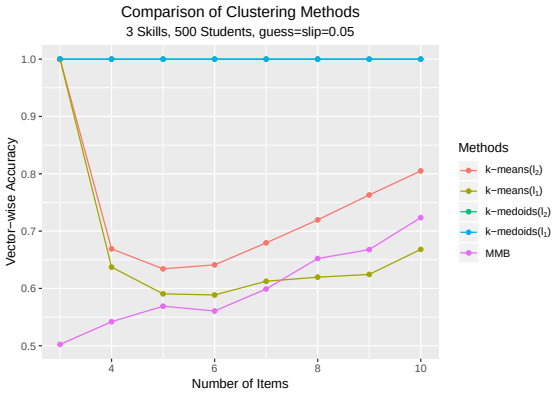


(f) Vector-wise Accuracy for guess=slip=0.2

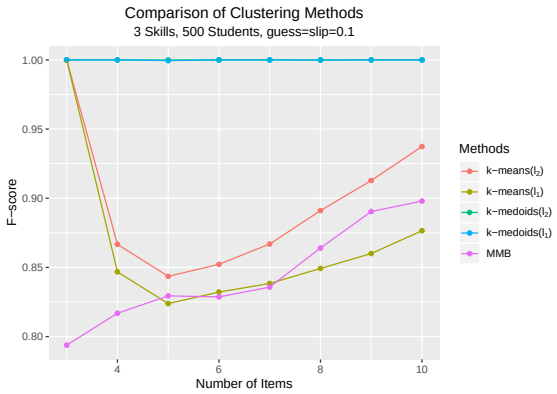
Figure 5.1 Clustering Method Comparison on k=3, N=100



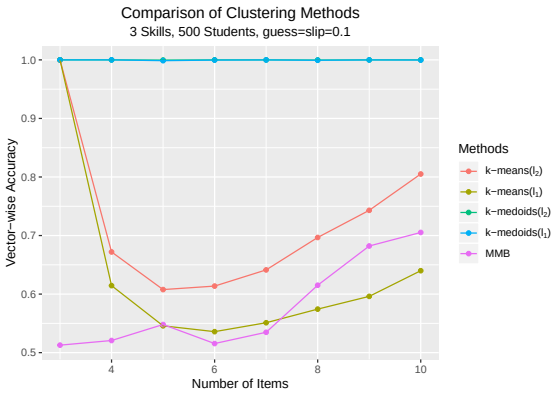
(a) F-score for cell-wise guess=slip=0.05



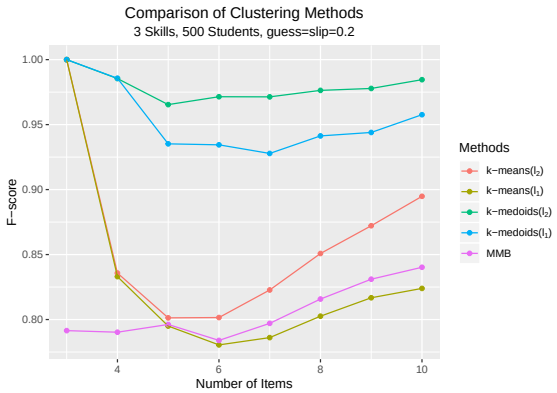
(b) Vector-wise Accuracy for guess=slip=0.05



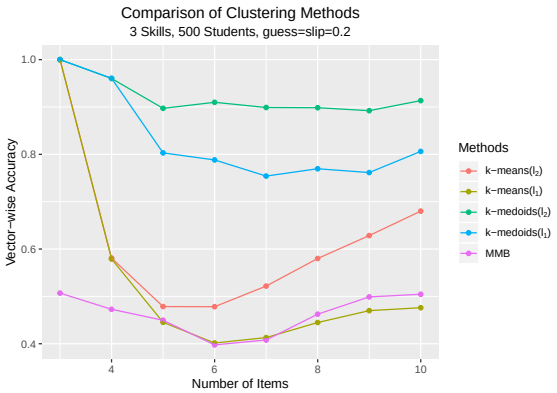
(c) F-score for cell-wise guess=slip=0.1



(d) Vector-wise Accuracy for guess=slip=0.1

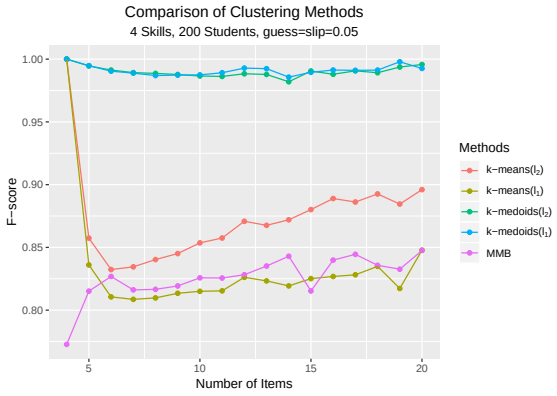


(e) F-score for cell-wise guess=slip=0.2

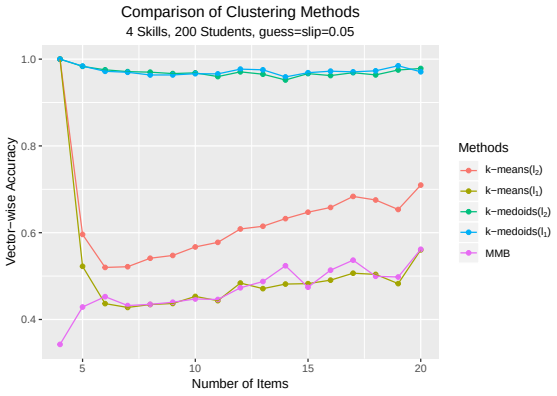


(f) Vector-wise Accuracy for guess=slip=0.2

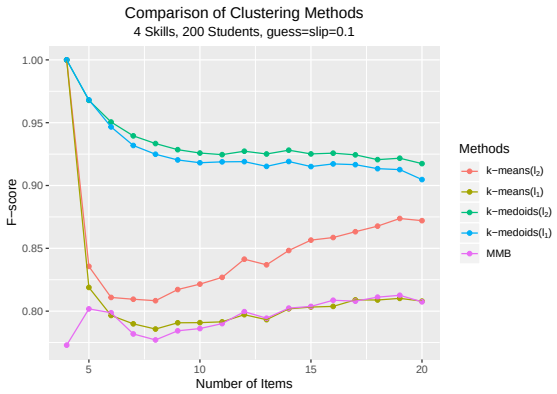
Figure 5.2 Clustering Method Comparison on k=3, N=500



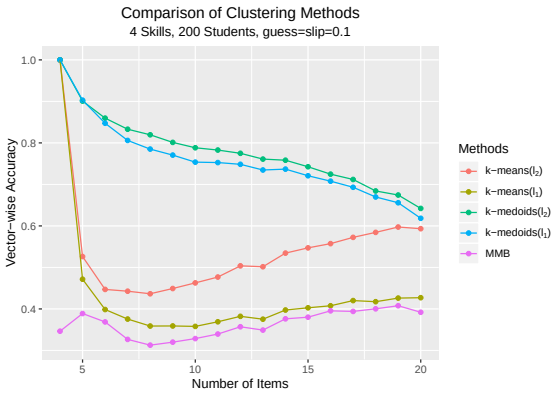
(a) F-score for cell-wise guess=slip=0.05



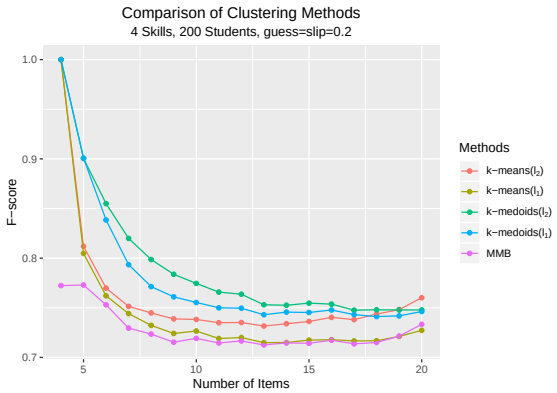
(b) Vector-wise Accuracy for guess=slip=0.05



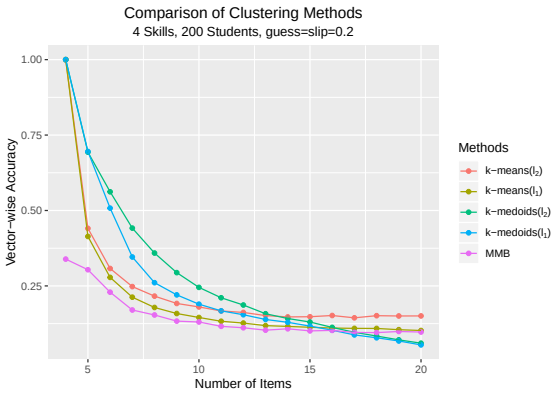
(c) F-score for cell-wise guess=slip=0.1



(d) Vector-wise Accuracy for guess=slip=0.1

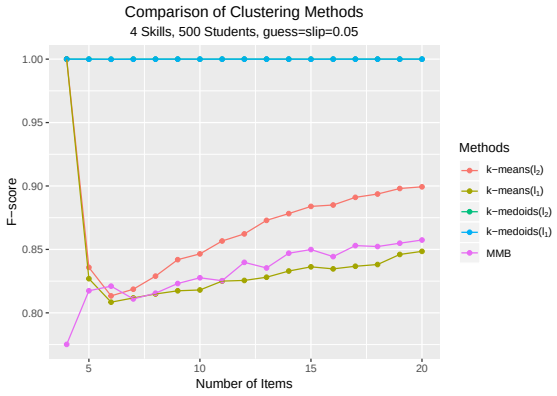


(e) F-score for cell-wise guess=slip=0.2

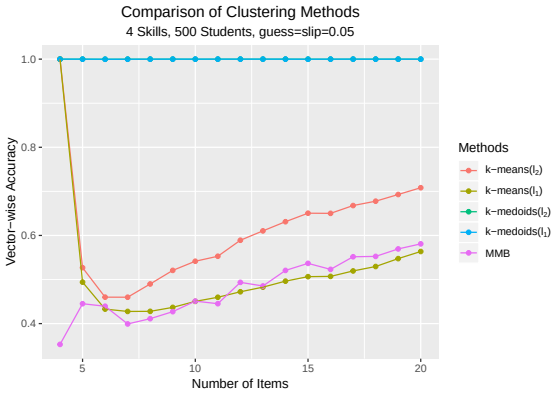


(f) Vector-wise Accuracy for guess=slip=0.2

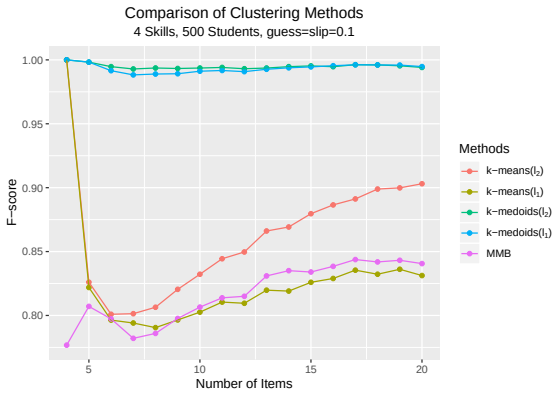
Figure 5.3 Clustering Method Comparison on k=4, N=200



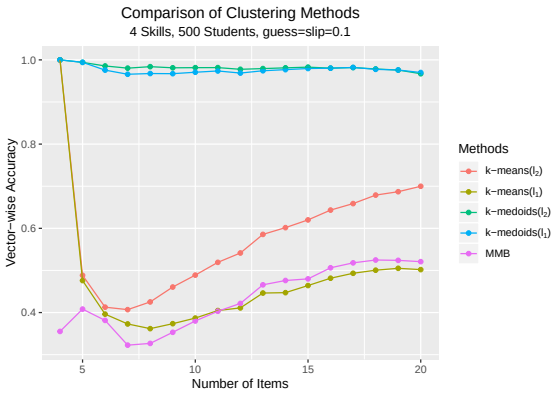
(a) F-score for cell-wise guess=slip=0.05



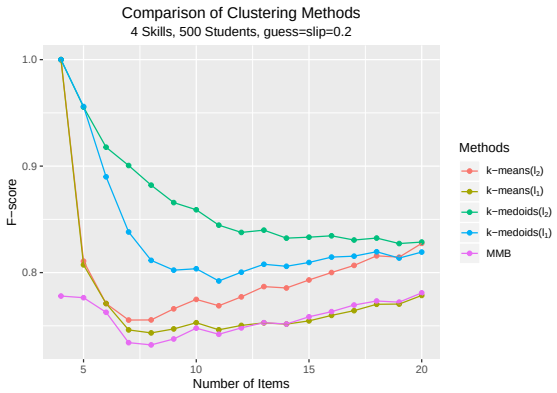
(b) Vector-wise Accuracy for guess=slip=0.05



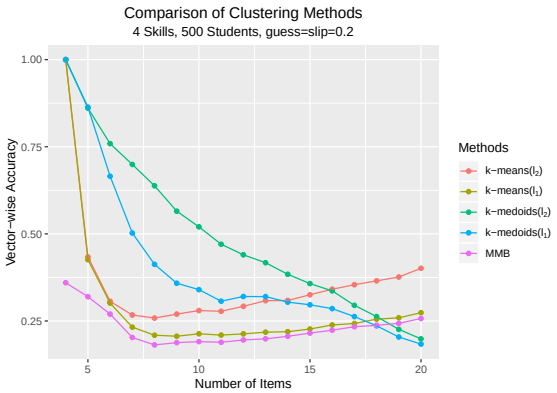
(c) F-score for cell-wise guess=slip=0.1



(d) Vector-wise Accuracy for guess=slip=0.1



(e) F-score for cell-wise guess=slip=0.2



(f) Vector-wise Accuracy for guess=slip=0.2

Figure 5.4 Clustering Method Comparison on k=4, N=500

From the figures we can also see that the k-medoids using ℓ_2 distance performs best almost across all configurations, except when slip and guess are high, then k-means using ℓ_2 distance might surpass k-medoids in case of large number of items (Figures 5.3f, 5.4f).

5.6.2 IRPtoQ: A Nonparametric Method to derive Q-matrix

After having obtained the IRPs from clustering, now we can derive Q-matrix from them. It is obvious that each IRP is associated with a student profile pattern, but how can we find the desired Q-matrix from it?

Let us observe an example first. Consider a Q-matrix

$$Q = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{5.9}$$

then the correspondence between all profiles and their ideal response patterns correspondence under DINA model can be calculated and written out in table 5.1, in which we ordered them increasingly by the row sum.

Table 5.1 Profile-IRP correspondence example

	Profiles			Ideal Response Patterns									
	s1	s2	s3	i1	i2	i3	i4	i5	i6	i7	i8	i9	row sum
pr_1	0	0	0	0	0	0	0	0	0	0	0	0	0
pr_2	1	0	0	0	0	0	1	0	0	0	0	0	1
pr_3	0	1	0	0	0	0	0	0	1	0	0	0	1
pr_4	0	0	1	0	0	0	0	1	0	0	0	0	1
pr_5	1	1	0	1	0	0	1	0	1	0	0	0	3
pr_6	1	0	1	0	0	1	1	1	0	0	0	0	3
pr_7	0	1	1	0	1	0	0	1	1	0	1	1	5
pr_8	1	1	1	1	1	1	1	1	1	1	1	1	9

Our hypothesis is we can get the correct IRP matrix through clustering, which looks like the ideal response patterns in Table 5.1, under some row exchanges. Moreover, if we reorder our clustering result increasingly by row sum, we should obtain the IRP matrix looking exactly like the one in the Table 5.1 (there might still be some exceptions due to the same number of some row sums, just like the row 2-4 in our example). The reason is, for a two-skill profile like $pr_5 = (1, 1, 0)$, the row sum of its IRP should always be bigger than the row sum of the IRPs of one-skill profiles $pr_2 = (1, 0, 0)$ and $pr_3 = (0, 1, 0)$, since if a student masters skill 1 and 2, one also should always answer those questions involving only skill 1 or 2 correctly. However, we point out that this is not true for a third skill. For example, the row sum of the IRP of the profile $pr_5 = (1, 1, 0)$ is not necessarily bigger than the row sum of the IRP of the profile $pr_4 = (0, 0, 1)$. The reason is we might have a lot of items requiring the skill 3 solely, while only a few items requiring skill 1 or skill 2 or both. If the sum of the numbers of items of all these three cases (only requiring skill 1 or skill 2 or both) is still smaller than number of items requiring only skill 3, then we will have the row sum of the IRP of a one-skill profile $pr_4 = (0, 0, 1)$ being bigger than the one of two-skill profile $pr_5 = (1, 1, 0)$. Nevertheless, practically speaking that would be a rare case, since it means we will have a test of items that extremely focus on one particular skill, and the number of which even surpasses that of the items diagnosing all other skills.

Therefore, if we denote the number of items requiring item i by c_i, then we can make an assumption that different c_i does not differ that much. We will call this assumption *evenness* hypothesis in this thesis. In fact, we can see that c_i corresponds to the sum of the i-th column of the Q-matrix. Thus in the above example we have $c_1 = 4$, $c_2 = 6$, $c_3 = 6$, which would not cause the problem we described above.

Subsequently, with the evenness hypothesis, together with the separability hypothesis for the clustering (all possible profiles are present and distinguishable by clustering, except the no-skill-mastered profile is allowed to miss), we can start to find the correspondent profiles of all our obtained IRPs, which helps find out the q-vectors for each item. For example, we can write down the IRPs obtained from some clustering method after reordering in Table 5.2.

Table 5.2 IRPs derived from clustering algorithm

	Profiles			Ideal Response Patterns from Clustering									
	s1	s2	s3	i1	i2	i3	i4	i5	i6	i7	i8	i9	row sum
pr_1	0	0	0	0	0	0	0	0	0	0	0	0	0
pr_2	1	0	0	0	0	0	1	0	0	0	0	0	1
pr_3	0	1	0	0	0	0	0	1	0	0	0	0	1
pr_4	0	0	1	0	0	0	0	0	1	0	0	0	1
pr_5	1	1	0	0	0	1	1	1	0	0	0	0	3
pr_6	1	0	1	1	0	0	1	0	1	0	0	0	3
pr_7	0	1	1	0	1	0	0	1	1	0	1	1	5
pr_8	1	1	1	1	1	1	1	1	1	1	1	1	9

We can see that the differences are that row 3 and row 4 are exchanged, row 5 and row 6 are also exchanged. Then we know that row 2-4 corresponds to 1-skill profile, row 5-7 corresponds to 2-skill profile and row 8 corresponds to 3-skill profile. For row 2-4, we can easily assign a different 1-skill profile for each of them, and the order of them does not matter. Afterwards, we get to know the q-vectors of item 4, 5 and 6 since we know now that they only require 1 skill. If we assign profile $(1,0,0)$, $(0,1,0)$, $(0,0,1)$ to IRP_2, IRP_3, IRP_4 respectively, then we can write our estimated q-vectors for items 4, 5, 6 to be $q_4 = (1,0,0)$, $q_5 = (0,1,0)$ and $q_6 = (0,0,1)$. For row 5-7, we need to compare each of them with the sum of two one-skill IRPs. That is, we calculate the sum of vectors IRP_2 and IRP_3, the sum of vectors IRP_2 and IRP_4 and the sum of vectors IRP_3 and IRP_4, yielding IRP vectors $IRP_{2,3} = (0,0,0,1,0,1,0,0,0)$, $IRP_{2,4} = (0,0,0,1,1,0,0,0,0)$ and $IRP_{3,4} = (0,0,0,0,1,1,0,0,0)$. Then for IRP_5, we compare it with all three of them. If we find an IRP vector $IRP_{i,j}$ satisfying the partial order constraint $IRP_5 \succcurlyeq IRP_{i,j}$, meaning each cell of IRP_5 is no less than its correspondence in vector $IRP_{i,j}$, then IRP_5 is correspondent to profile that masters skill i and skill j. In our case, $IRP_5 \succcurlyeq IRP_{2,3}$, thus IRP_5 corresponds to profile $(1,1,0)$ and the non-zero items in IRP_5 which has not been assigned q-vectors (unlike item i4 and i5), i.e. i3, is now assigned with q-vector $q_3 = (1,1,0)$. This calculation proceeds for all the remaining 2-skill items and also 3-skill items. After exhausting all IRPs, we will obtain all their correspondent profiles as well as q-vectors as byproducts. Finally, for our

example, we obtain an estimated Q-matrix

$$\hat{Q} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{5.10}$$

We can see that it is equivalent to the true Q-matrix in 5.9, since the only difference is that the former is a column 2-3 exchanged version of the latter.

We summarize the whole procedure described above in Algorithm 3.

Algorithm 3: IRPtoQ Algorithm

input : A K by J Matrix M representing ideal response patterns (IRPs) learned from clustering, latent number of skills k, notice $K = 2^k$

output: Q-matrix

1. Order the rows of the matrix M increasingly based on row sums;
2. Ignoring the first row, and choose the first k rows as the IRPs for one-skill profiles;
3. Remove conflicts;
4. Assign one-skill q-vectors to the non-zero items for the first k IRPs;
5. **for** $i \leftarrow 2$ **to** k **do**
   a. Compare i-skill IRPs with combinations of one-skill IRPs, and choose the one satisfying the partial order constraint the most;
   b. Assign q-vector for i-skill case items

end

5. **return** (Q-matrix)

The step 3 is to remove conflicts, which is a practical situation we have to deal with. Because of slip and guess, we do not always get completely correct IRPs. Ideally for the one-skill IRP, there should not be any item that appears to be 1 in two or more one-skill IRPs, since that means it is not a one-skill item which contradicts to being 1 in a one-skill IRP. Therefore we add the step 3 of conflict removal, which changes the entries of an item that shows to be 1 across multiple IRPs to be 0 starting from the second time of show-up. For example, if the

first three IRP we obtain are

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Then the item 4 has conflicts, since if we assign skill 1,2,3 to each of them, then item 4 requires both skill 1 and 3, which is incorrect for one-skill IRP case. Since these rows are ordered by row sum, we always favour to solve conflicts on later rows. Therefore, here we set the entry of item 4 in the third row to be 0, after that, the first three IRPs become

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and the conflict is removed.

Similarly for step 5, we might not find a combination of one-skill IRPs that completely satisfying the partial order constraint, thus we have to choose the one that has the least conflicts, meaning it has the most cells that satisfying the constraint. For example, if the first three IRPs are:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

and the fourth IRP we have is:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Then the combination of the three one-skill IRP for two-skill IRPs are $IRP_{1,2} = (1, 0, 1, 1, 0, 1, 0, 0, 0)$, $IRP_{1,3} = (1, 0, 0, 1, 1, 0, 1, 0, 0)$, $IRP_{2,3} = (0, 0, 1, 0, 1, 1, 1, 0, 0)$, We can see that the conflicts (cells that in IRP_4 are smaller than those in the combination IRP) between IRP_4 and these three IRPs are $conflict_{1,2} = 1$, $conflict_{1,3} = 3$, $conflict_{2,3} = 2$, thus we deduce that IRP_4 corresponds to profile $(1, 1, 0)$ and the q-vector for item 8 is $q_8 = (1, 1, 0)$.

Experiments

Now we can compare our new IRPtoQ method with other Q-matrix derivation methods by experiments. Here we use the same experiment setup applied in the previous clustering experiments since we are using the IRPtoQ algorithm on their clustering results. Similarly

to the clustering methods comparison, we use three metrics to compare different Q-matrix derivation methods, which are cell-wise F-score, cell-wise accuracy and vector-wise accuracy. RMSE is no longer used here since all the Q-matrices considered are binary. The definitions of these metrics are the same as those used in clustering methods comparison, only except that here the vectors or cells are q-vectors or their entries, while in clustering methods comparison the vectors or cells are the IRPs or their entries.

Results

For conciseness, we only show IRPtoQ performance using k-medoids with ℓ_2 distance clustering results, since we have shown in Section 5.6.1 that it is the best performing method among all clustering methods. But IRPtoQ method using clustering results from other clustering methods are still reported in Appendix B for reference.

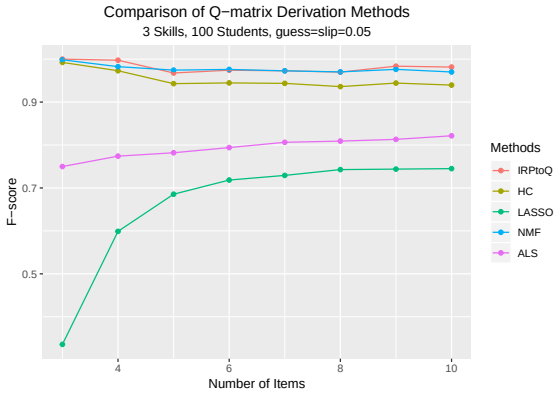
First, from the result we can see that even if the clustering techniques gives accurate ideal response patterns (Figure 5.1a), our algorithm of converting it to Q-matrix still does not yield a completely correct Q-matrix (Figure 5.5a). The reason can be illustrated from the example below. Consider a Q-matrix:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{5.11}$$

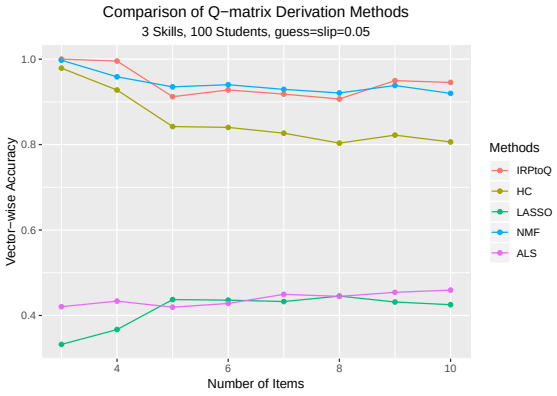
This Q-matrix in 5.11 is problematic for the IRPtoQ method, since it does not satisfy our "evenness" hypothesis. In fact, skill 1 is tested 3 times, more than the combination of skill 2 and 3. However, in our simulation, the generation of this kind of Q-matrices are permitted.

Second, we see that the IRPtoQ performance is highly dependent on the accuracy of clustering results. When IRPs obtained from clustering are accurate, then IRPtoQ will perform well, otherwise it would perform badly. This can be found by seeing the consistency between the correspondent figures (e.g. Figure 5.4a and 5.8a). In fact, for those good clustered cases, we can see that IRPtoQ generally performs well (Figure 5.7a, 5.7b, 5.8a, 5.8b, 5.8c, 5.8d).

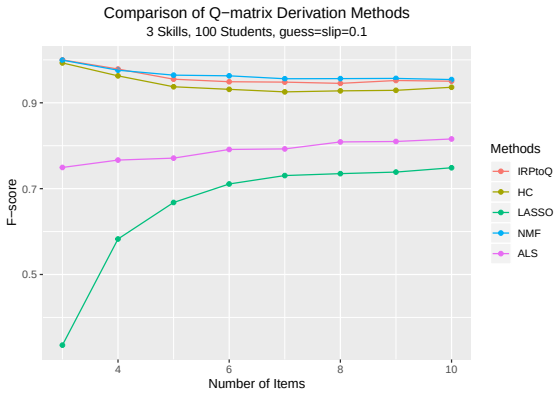
Third, we see that NMF is also a comparatively good method. Not only the performance of it is close to IRPtoQ in 3-skill case (Figures 5.5a, 5.5b), but also it can be more robust under some situations (e.g. Figures 5.7d, 5.7f).



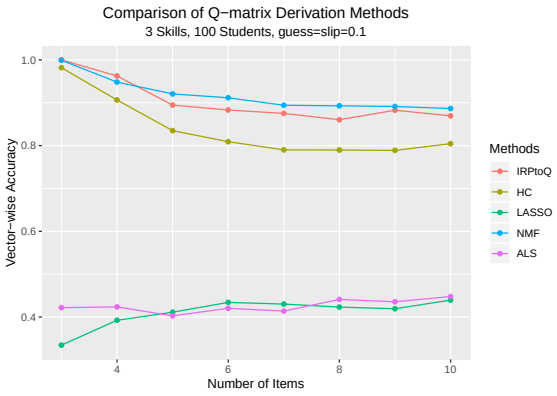
(a) F-score for cell-wise guess=slip=0.05



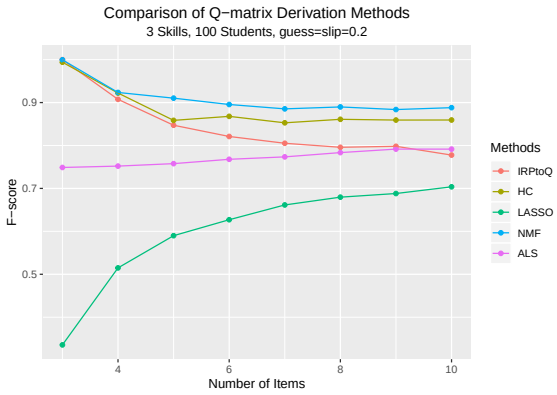
(b) Vector-wise Accuracy for guess=slip=0.05



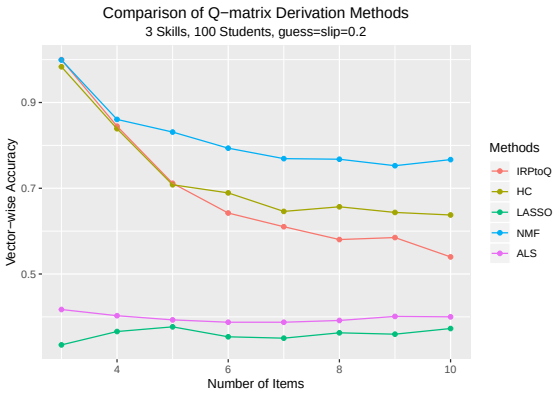
(c) F-score for cell-wise guess=slip=0.1



(d) Vector-wise Accuracy for guess=slip=0.1

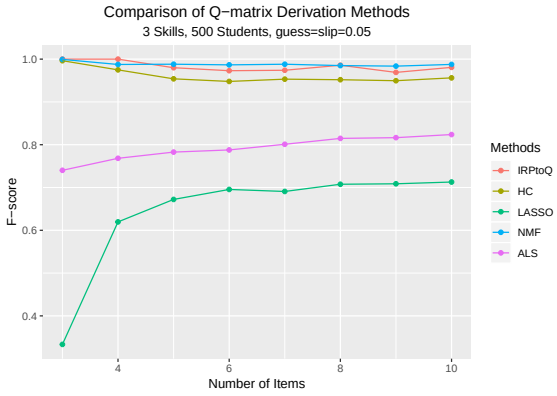


(e) F-score for cell-wise guess=slip=0.2

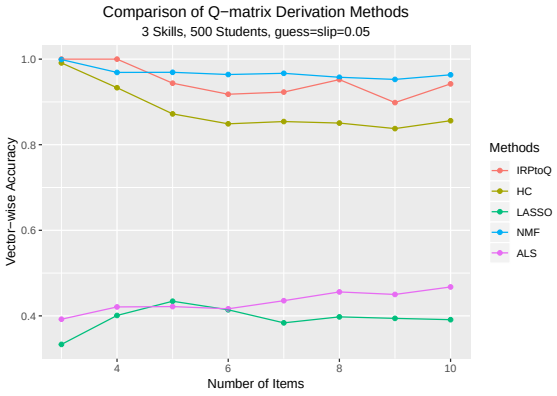


(f) Vector-wise Accuracy for guess=slip=0.2

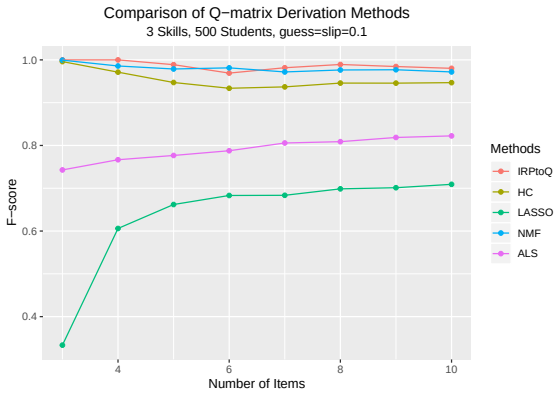
Figure 5.5 Derivation Method Comparison on k=3, N=100



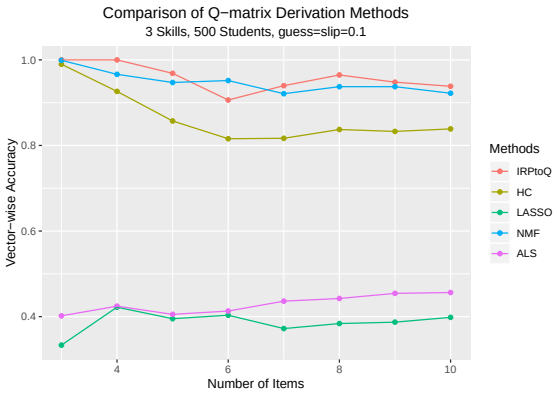
(a) F-score for cell-wise guess=slip=0.05



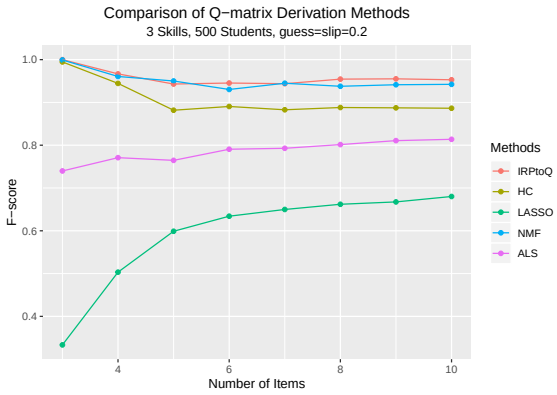
(b) Vector-wise Accuracy for guess=slip=0.05



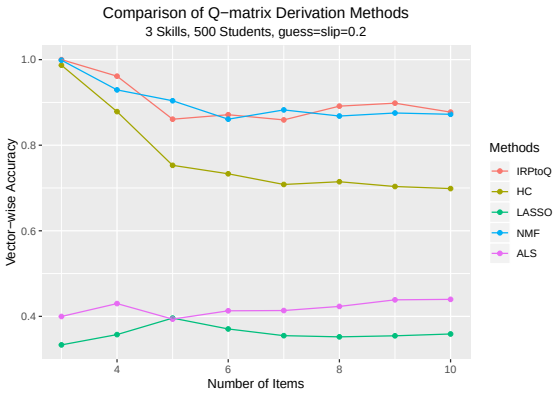
(c) F-score for cell-wise guess=slip=0.1



(d) Vector-wise Accuracy for guess=slip=0.1

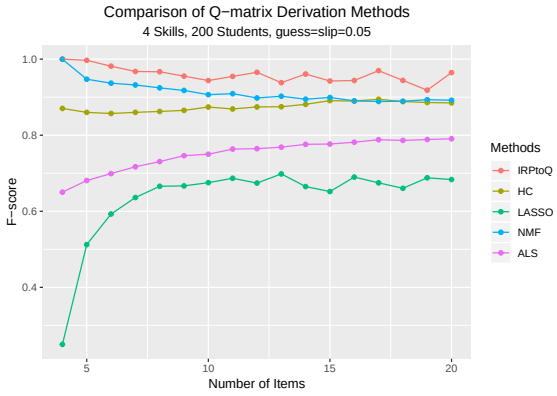


(e) F-score for cell-wise guess=slip=0.2

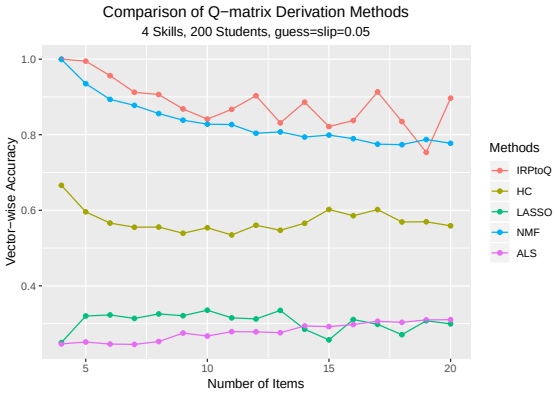


(f) Vector-wise Accuracy for guess=slip=0.2

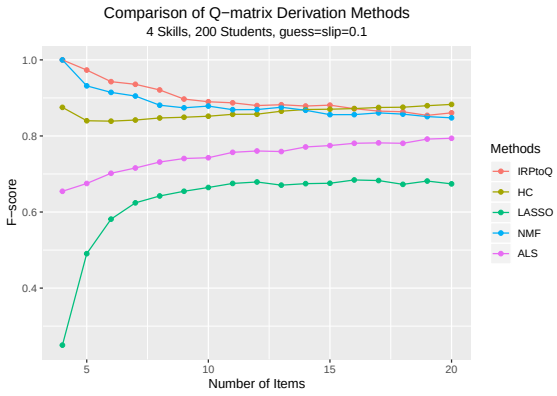
Figure 5.6 Derivation Method Comparison on k=3, N=500



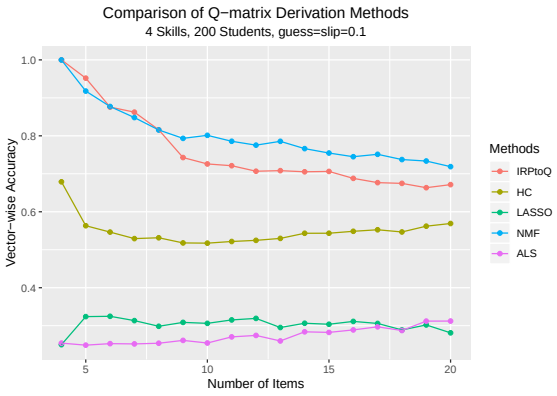
(a) F-score for cell-wise guess=slip=0.05



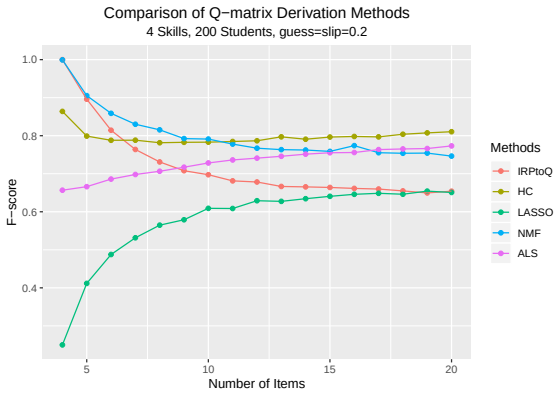
(b) Vector-wise Accuracy for guess=slip=0.05



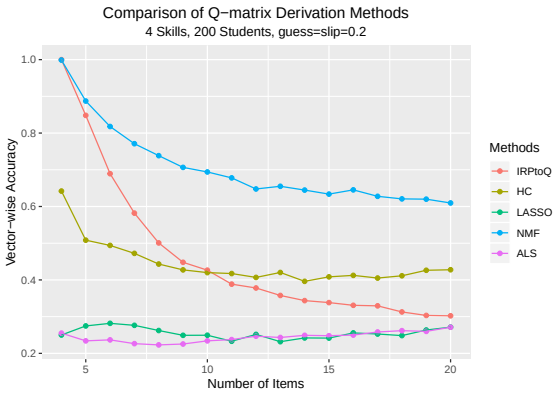
(c) F-score for cell-wise guess=slip=0.1



(d) Vector-wise Accuracy for guess=slip=0.1

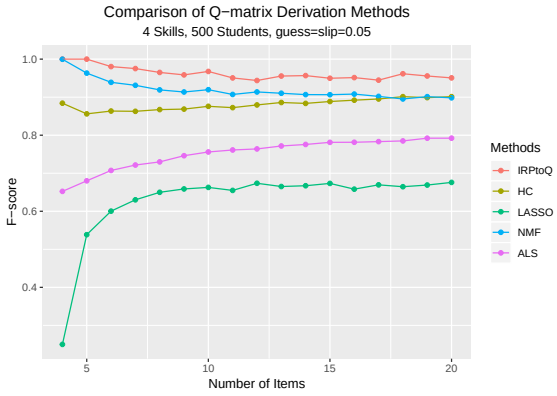


(e) F-score for cell-wise guess=slip=0.2

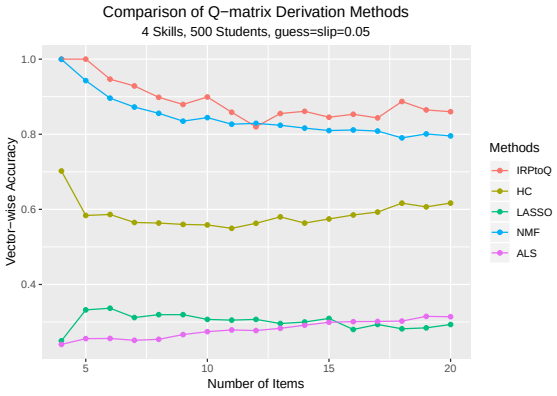


(f) Vector-wise Accuracy for guess=slip=0.2

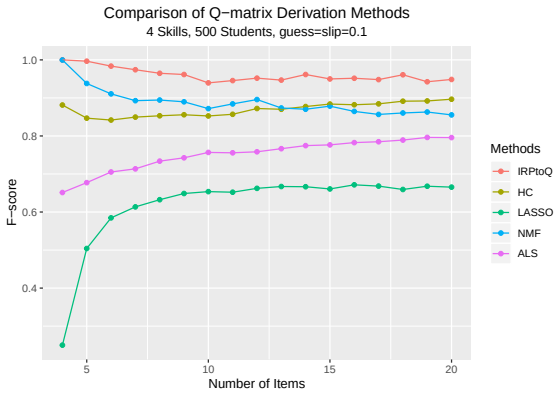
Figure 5.7 Derivation Method Comparison on k=4, N=200



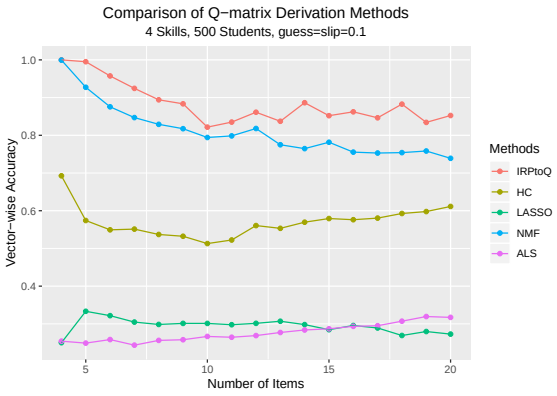
(a) F-score for cell-wise guess=slip=0.05



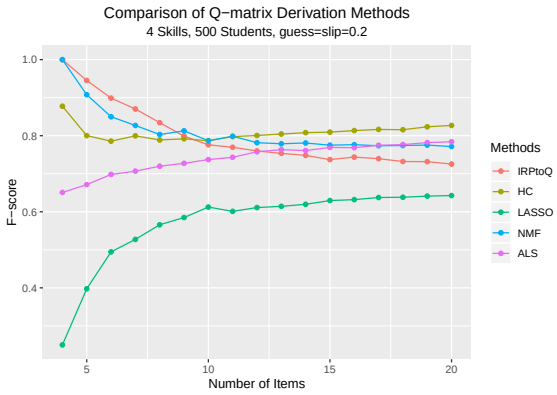
(b) Vector-wise Accuracy for guess=slip=0.05



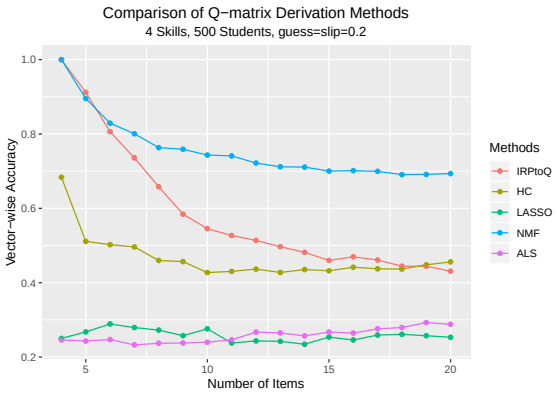
(c) F-score for cell-wise guess=slip=0.1



(d) Vector-wise Accuracy for guess=slip=0.1



(e) F-score for cell-wise guess=slip=0.2



(f) Vector-wise Accuracy for guess=slip=0.2

Figure 5.8 Derivation Method Comparison on k=4, N=500

CHAPTER 6 CONCLUSION

6.1 Summary of Works

We conducted studies in three topics, namely Q-matrix refinement, design and derivation. For Q-matrix refinement, we introduced an ensemble technique to combine multiple refinement models, and used a boosted decision tree algorithm to improve the performance. For Q-matrix design, we investigated the relationship between identifiability and Q-matrix design, and concludes that a good Q-matrix design is consists of single skill questions, rather than questions involving more skills. The claim is supported both empirically and theoretically. Last, we proposed a new method for Q-matrix derivation, which contrasted with current state-of-the-art methods, in the way that it is essentially nonparametric, which makes it easy to implement. The approach outperforms other algorithms in synthetic datasets.

The research in this thesis has its practical usage. In fact, techniques of Q-matrix refinement can help teachers better inspect expert-given or self-precified Q-matrix. Principles of Q-matrix design can help teachers devise better question set for efficient student profile diagnosis. And last, Q-matrix derivation algorithms can help teachers solve the "cold-start" problem, meaning when there is no expert-given Q-matrix at hand, we can still obtain a reasonable Q-matrix directly from student response data.

6.2 Limitations and future research

A first limitation of our research is its reliance on static data, which do not consider the time factor, or sequential data. It implicitly relies on the assumption that no learning occurs within the data. For online tutoring system that gather sequential data in time, this assumption is unrealistic. How much the violation of the assumption affects the result is a topic of future research, and new models that can handle dynamic student profiles will be required. The work of Koedinger, Stamper, McLaughlin, and Nixon (2013) is a good example of such developments.

A second limitation is that all of our research is based on the DINA model, which is an over-simplified model in some cases. This limitation may not be that significant for some of the findings in this thesis. For example, the Q-matrix refinement technique that learns from synthetic data and a boosted decision tree is likely to also work to improve over existing techniques that work with DINO or compensatory models, albeit with different levels of improvements, whether better or worse. The same conjecture may be asserted for the non

parametric Q-matrix derivation method proposed, although this is likely more speculative and would obviously need to be investigated empirically. However, the identifiability design principle would likely not hold, but more theoretical and empirical research would need to be conducted.

A particularly interesting and difficult case can be made for Q-matrices that are mixtures of conjunctive, disjunctive and compensatory principles. No research in that direction has yet been done, while it seems reasonable that some tasks can involve conjunction of skills as well as disjunctions. In the same whelm of research, Q-matrix with non binary data also appear to be a realistic relaxation of the binary constraint, albeit leading to more difficult human interpretation.

Finally, the research that use item features to create clusters and links between answers and questions, such as Goutte, Léger, and Durand (2015); Durand, Belacel, and Goutte (2016), is a highly promising and complementary approach to the work that relies on student performance data such as ours. There is an obvious gain to bridge and merge these two approaches and no work has been conducted in that direction yet.

REFERENCES

Aldous, D. J. (1985). Exchangeability and related topics. In *École d'été de probabilités de saint-flour xiii—1983* (pp. 1–198). Springer.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, *43*(4), 561–573.

Andrich, D. (1988). *Rasch models for measurement* (Vol. 68). Sage.

Antoniak, C. E. (1974). Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics*, 1152–1174.

Austerweil, J. L., Gershman, S. J., Tenenbaum, J. B., & Griffiths, T. L. (2015). Structure and flexibility in bayesian models of cognition. *Oxford handbook of computational and mathematical psychology*, 187–208.

Barnes, T. (2010). Novel derivation and application of skill matrices: The Q-matrix method. *Handbook on educational data mining*, 159–172.

Bartholomew, D. J., Knott, M., & Moustaki, I. (2011). *Latent variable models and factor analysis: A unified approach* (Vol. 904). John Wiley & Sons.

Bishop, C. M. (1994). *Mixture density networks* (Tech. Rep.). Citeseer.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Breiman, L. (1999). Combining predictors. *Combining artificial neural nets*, *31*.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Brewer, P. W. (1996). *Methods for concept mapping in computer based education* (Unpublished doctoral dissertation). North Carolina State University.

Casella, G., & Berger, R. L. (2002). *Statistical inference* (Vol. 2). Duxbury Pacific Grove, CA.

Chen, Y., Culpepper, S. A., Chen, Y., & Douglas, J. (2018). Bayesian estimation of the DINA Q-matrix. *Psychometrika*, *83*(1), 89–108.

Chen, Y., Li, X., Liu, J., & Ying, Z. (2017). Regularized latent class analysis with application in cognitive diagnosis. *psychometrika*, *82*(3), 660–692.

Chen, Y., Liu, J., Xu, G., & Ying, Z. (2015). Statistical analysis of Q-matrix based diagnostic classification models. *Journal of the American Statistical Association*, *110*(510), 850–866.

Chiu, C.-Y. (2013). Statistical refinement of the Q-matrix in cognitive diagnosis. *Applied Psychological Measurement*, *37*(8), 598–618.

Chiu, C.-Y., & Douglas, J. (2013). A nonparametric approach to cognitive diagnosis by proximity to ideal response patterns. *Journal of Classification*, *30*(2), 225–250.

Chiu, C.-Y., Douglas, J. A., & Li, X. (2009). Cluster analysis for cognitive diagnosis: Theory and applications. *Psychometrika*, *74*(4), 633.

Culpepper, S. A. (2015). Bayesian estimation of the DINA model with gibbs sampling. *Journal of Educational and Behavioral Statistics*, *40*(5), 454–476.

da Silva, M. A., de Oliveira, E. S., von Davier, A. A., & Bazán, J. L. (2018). Estimating the DINA model parameters using the no-u-turn sampler. *Biometrical Journal*, *60*(2), 352–368.

DeCarlo, L. T. (2012). Recognizing uncertainty in the q-matrix via a bayesian extension of the dina model. *Applied Psychological Measurement*, *36*(6), 447–468.

de Fréin, R., Drakakis, K., Rickard, S., & Cichocki, A. (2008). Analysis of financial data using non-negative matrix factorization. In *International mathematical forum* (Vol. 3, pp. 1853–1870).

De La Torre, J. (2008). An empirically based method of Q-matrix validation for the DINA model: Development and applications. *Journal of educational measurement*, *45*(4), 343–362.

De La Torre, J. (2009). DINA model and parameter estimation: A didactic. *Journal of Educational and Behavioral Statistics*, *34*(1), 115–130.

De La Torre, J., & Douglas, J. A. (2004). Higher-order latent trait models for cognitive diagnosis. *Psychometrika*, *69*(3), 333–353.

De Menezes, L. (1999). On fitting latent class models for binary data: The estimation of standard errors. *British Journal of Mathematical and Statistical Psychology*, *52*(2), 149–168.

Desmarais, M. (2011). Conditions for effectively deriving a Q-matrix from data with non-negative matrix factorization. In *4th international conference on educational data mining, edm2014* (pp. 41–50).

Desmarais, M. C. (2012). Mapping question items to skills with non-negative matrix factorization. *ACM SIGKDD Explorations Newsletter*, *13*(2), 30–36.

Desmarais, M. C., Beheshti, B., & Naceur, R. (2012). Item to skills mapping: deriving a conjunctive Q-matrix from data. In *International conference on intelligent tutoring systems* (pp. 454–463).

Desmarais, M. C., & Naceur, R. (2013). A matrix factorization method for mapping items to skills and for enhancing expert-based q-matrices. In *Artificial intelligence in education, AIED2013* (pp. 441–450).

Desmarais, M. C., Xu, P., & Beheshti, B. (2015). Combining techniques to refine item to skills q-matrices with a partition tree. In *Proceedings of the 8th educational data mining conference, EDM2015.*

DiBello, L. V., Roussos, L. A., & Stout, W. (2006). 31a review of cognitively diagnostic assessment and a summary of psychometric models. *Handbook of statistics*, *26*, 979–1030.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, *195*(2), 216–222.

Durand, G., Belacel, N., & Goutte, C. (2016). Competency based learning in the web of learning data. In *Proceedings of the 25th international conference companion on world wide web* (pp. 489–494).

Embretson, S. E., & Reise, S. P. (2013). *Item response theory.* Psychology Press.

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, *96*(456), 1348–1360.

Fan, J., & Lv, J. (2011). Nonconcave penalized likelihood with np-dimensionality. *IEEE Transactions on Information Theory*, *57*(8), 5467–5484.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and computation*, *121*(2), 256–285.

Freund, Y., & Schapire, R. E. (1996a). Experiments with a new boosting algorithm. In *icml* (Vol. 96, pp. 148–156).

Freund, Y., & Schapire, R. E. (1996b). Game theory, on-line prediction and boosting. In *Colt* (Vol. 96, pp. 325–332).

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, *28*(2), 337–407.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, *38*(4), 367–378.

Gershman, S. J., & Blei, D. M. (2012). A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, *56*(1), 1–12.

Goutte, C., Léger, S., & Durand, G. (2015). A probabilistic model for knowledge component naming. In *Edm* (pp. 608–609).

Grant, R. L., Furr, D. C., Carpenter, B., & Gelman, A. (2017). Fitting bayesian item response models in stata and stan. *The Stata Journal*, *17*(2), 343–357.

Gupta, M. D., & Xiao, J. (2011). Non-negative matrix factorization as a feature selection tool for maximum margin classifiers. In *Cvpr 2011* (pp. 2841–2848).

Haertel, E. H. (1989). Using restricted latent class models to map the skill structure of achievement items. *Journal of Educational Measurement*, 301–321.

Hagenaars, J. A. (1993). *Loglinear models with latent variables* (No. 94). Sage.

Hastie, T., Rosset, S., Zhu, J., & Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface*, *2*(3), 349–360.

Henson, R. A., Templin, J. L., & Willse, J. T. (2009). Defining a family of cognitive diagnosis models using log-linear models with latent variables. *Psychometrika*, *74*(2), 191–210.

Hjort, N. L., Holmes, C., Müller, P., & Walker, S. G. (2010). *Bayesian nonparametrics* (Vol. 28). Cambridge University Press.

Hoffman, M. D., & Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, *15*(1), 1593–1623.

Juan, A., & Vidal, E. (2002). On the use of bernoulli mixture models for text classification. *Pattern Recognition*, *35*(12), 2705–2710.

Juan, A., & Vidal, E. (2004). Bernoulli mixture models for binary images. In *Pattern recognition, 2004. icpr 2004. proceedings of the 17th international conference on* (Vol. 3, pp. 367–370).

Kearns, M. J., Vazirani, U. V., & Vazirani, U. (1994). *An introduction to computational learning theory*. MIT press.

Kendall, M., Stuart, A., Ord, J., & Arnold, S. (1999). Vol. 2a: Classical inference and the linear model. *London [etc.]: Arnold [etc.]*.

Koedinger, K. R., Stamper, J. C., McLaughlin, E. A., & Nixon, T. (2013). Using data-driven discovery of better student models to improve student learning. In *International conference on artificial intelligence in education* (pp. 421–430).

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788–791.

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems* (pp. 556–562).

Li, F. (2008). *A modified higher-order dina model for detecting differential item functioning and differential attribute functioning* (Unpublished doctoral dissertation). uga.

Lindsey, R. V., Khajah, M., & Mozer, M. C. (2014). Automatic discovery of cognitive skills to improve the prediction of student learning. In *Advances in neural information processing systems* (pp. 1386–1394).

Liu, J., Xu, G., & Ying, Z. (2013). Theory of the self-learning Q-matrix. *Bernoulli: official journal of the Bernoulli Society for Mathematical Statistics and Probability*, *19*(5A), 1790.

Lord, F. M. (2012). *Applications of item response theory to practical testing problems*. Routledge.

Lord, F. M., & Novick, M. R. (2008). *Statistical theories of mental test scores*. IAP.

Maris, E. (1999). Estimating multiple classification latent class models. *Psychometrika*, *64*(2), 187–212.

McLachlan, G. J. (1988). Mixture models: Inference and applications to clustering. *Inference and Applications to Clustering.*.

Meir, R., & Rätsch, G. (2003). An introduction to boosting and leveraging. In *Advanced lectures on machine learning* (pp. 118–183). Springer.

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal processing magazine*, *13*(6), 47–60.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nugroho, D. B., & Morimoto, T. (2015). Estimation of realized stochastic volatility models using hamiltonian monte carlo-based methods. *Computational Statistics*, *30*(2), 491–516.

Paatero, P. (1997). Least squares formulation of robust non-negative factor analysis. *Chemometrics and intelligent laboratory systems*, *37*(1), 23–35.

Paatero, P., & Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, *5*(2), 111–126.

Park, H.-S., & Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, *36*(2), 3336–3341.

Penrose, R. (1956). On best approximate solutions of linear matrix equations. In *Mathematical proceedings of the cambridge philosophical society* (Vol. 52, pp. 17–19).

Phelps, E. B. (1996). *Factor rank of boolean matrices* (Unpublished doctoral dissertation). University of Colorado at Denver.

Pitman, J. (2002). *Combinatorial stochastic processes* (Tech. Rep.). Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes for ....

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Quinlan, J. R. (1993). *C4. 5: Programs for machine learning* (Vol. 1). Morgan Kaufmann.

Rasch, G. (1960). Studies in mathematical psychology: I. probabilistic models for some intelligence and attainment tests.

Robitzsch, A., Kiefer, T., George, A. C., & Ünlü, A. (2017). Cdm: Cognitive diagnosis modeling [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=CDM` (R package version 5.4-0)

Roussos, L. A., Templin, J. L., & Henson, R. A. (2007). Skills diagnosis using IRT-based latent class models. *Journal of Educational Measurement*, *44*(4), 293–311.

Rudin, C., Daubechies, I., & Schapire, R. E. (2004). The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, *5*(Dec), 1557–1595.

Rupp, A. A., & Templin, J. (2008). The effects of Q-matrix misspecification on parameter estimates and classification accuracy in the DINA model. *Educational and Psychological Measurement*, *68*(1), 78–96.

Rupp, A. A., Templin, J., & Henson, R. A. (2010). *Diagnostic measurement: Theory, methods, and applications*. Guilford Press.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, *5*(2), 197–227.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine learning*, *37*(3), 297–336.

Shahnaz, F., Berry, M. W., Pauca, V. P., & Plemmons, R. J. (2006). Document clustering using nonnegative matrix factorization. *Information Processing & Management*, *42*(2), 373–386.

Sun, Y., Ye, S., Inoue, S., & Sun, Y. (2014). Alternating recursive method for Q-matrix learning. In *Educational data mining 2014*.

Tatsuoka, K. K. (1983). Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of educational measurement*, *20*(4), 345–354.

Tatsuoka, K. K. (1984). *Analysis of errors in fraction addition and subtraction problems*. Computer-based Education Research Laboratory, University of Illinois.

Tatsuoka, K. K. (1990). Toward an integration of item-response theory and cognitive error diagnosis. *Diagnostic monitoring of skill and knowledge acquisition*, 453–488.

Tatsuoka, K. K. (1995). Architecture of knowledge structures and cognitive diagnosis: A statistical pattern recognition and classification approach. *Cognitively diagnostic assessment*, 327–359.

Tatsuoka, K. K. (2009). *Cognitive assessment: An introduction to the rule space method*. Routledge.

Templin, J. L., & Henson, R. A. (2006). Measurement of psychological disorders using cognitive diagnosis models. *Psychological methods*, *11*(3), 287.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Valiant, L. G. (1984). A theory of the learnable. In *Proceedings of the sixteenth annual acm symposium on theory of computing* (pp. 436–445).

Wang, Y.-X., & Zhang, Y.-J. (2013). Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, *25*(6), 1336–1353.

Wasserman, L. (2006). *All of nonparametric statistics*. Springer Science & Business Media.

Xu, G. (2017). Identifiability of restricted latent class models with binary responses. *The Annals of Statistics*, *45*(2), 675–707.

Xu, G., & Shang, Z. (2018). Identifying latent structures in restricted latent class models. *Journal of the American Statistical Association*, *113*(523), 1284–1295.

Xu, G., & Zhang, S. (2015). Identifiability of diagnostic classification models. *Psychometrika*, 1–25.

Xu, P., & Desmarais, M. (2016). Boosted decision tree for Q-matrix refinement. In *Proceedings of the 9th educational data mining conference, EDM2016* (pp. 551–555).

Xu, P., & Desmarais, M. C. (2018). An empirical research on identifiability and Q-matrix design for DINA model. In *Proceedings of the 11th educational data mining conference, EDM2018* (pp. 438–444).

Xu, W., Liu, X., & Gong, Y. (2003). Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on research and development in informaion retrieval* (pp. 267–273).

Zhao, P., & Yu, B. (2006). On model selection consistency of Lasso. *Journal of Machine learning research*, *7*(Nov), 2541–2563.

APPENDIX A EXPERIMENT RESULTS OF CLUSTERING

A.1 Results of Clustering for IRPtoQ: 3-skill case

A.1.1 N=100, slip=guess=0.05

Table A.1 RMSE of clustering, k=3, N=100, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	0.0000	0.0000	0.0000	0.0000	0.1551
4	0.0838	0.1170	0.0041	0.0037	0.1218
5	0.0865	0.1285	0.0042	0.0045	0.1155
6	0.0829	0.1206	0.0051	0.0047	0.1104
7	0.0705	0.1159	0.0028	0.0031	0.1003
8	0.0625	0.1147	0.0031	0.0029	0.1001
9	0.0536	0.1060	0.0026	0.0019	0.0882
10	0.0517	0.1124	0.0030	0.0020	0.0851

Table A.2 F-score of clustering, k=3, N=100, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7736
4	0.8943	0.8603	0.9956	0.9960	0.8237
5	0.8895	0.8399	0.9954	0.9951	0.8474
6	0.8910	0.8438	0.9940	0.9945	0.8557
7	0.9055	0.8471	0.9967	0.9963	0.8675
8	0.9174	0.8481	0.9962	0.9964	0.8690
9	0.9288	0.8574	0.9967	0.9976	0.8820
10	0.9317	0.8506	0.9961	0.9974	0.8866

Table A.3 Cell-wise accuracy of clustering, k=3, N=100, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7750
4	0.9028	0.8704	0.9959	0.9963	0.8379
5	0.9008	0.8550	0.9958	0.9955	0.8597
6	0.9059	0.8635	0.9949	0.9953	0.8734
7	0.9215	0.8700	0.9972	0.9969	0.8878
8	0.9319	0.8720	0.9969	0.9971	0.8896
9	0.9434	0.8834	0.9974	0.9981	0.9045
10	0.9459	0.8781	0.9970	0.9980	0.9082

Table A.4 Vector-wise accuracy of clustering, k=3, N=100, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.4583
4	0.7530	0.6545	0.9891	0.9901	0.5393
5	0.7207	0.5868	0.9874	0.9862	0.5785
6	0.7194	0.5884	0.9850	0.9862	0.6087
7	0.7544	0.5915	0.9896	0.9890	0.6349
8	0.7760	0.5896	0.9892	0.9894	0.6365
9	0.8040	0.6150	0.9904	0.9922	0.6740
10	0.8100	0.6105	0.9893	0.9915	0.6801

A.1.2 N=100, slip=guess=0.1

Table A.5 RMSE of clustering, N=100, slip=guess=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	0.0000	0.0000	0.0000	0.0000	0.1543
4	0.1064	0.1382	0.0232	0.0225	0.1322
5	0.1086	0.1360	0.0334	0.0375	0.1387
6	0.1025	0.1312	0.0351	0.0378	0.1371
7	0.0921	0.1263	0.0330	0.0350	0.1271
8	0.0800	0.1223	0.0266	0.0244	0.1181
9	0.0747	0.1194	0.0251	0.0238	0.1073
10	0.0681	0.1142	0.0238	0.0204	0.0984

Table A.6 F-score of clustering, k=3, N=100, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7768
4	0.8686	0.8336	0.9751	0.9758	0.8071
5	0.8577	0.8238	0.9628	0.9587	0.8122
6	0.8633	0.8238	0.9600	0.9571	0.8179
7	0.8775	0.8259	0.9602	0.9585	0.8294
8	0.8927	0.8278	0.9669	0.9699	0.8380
9	0.9015	0.8327	0.9684	0.9702	0.8530
10	0.9125	0.8403	0.9698	0.9742	0.8664

Table A.7 Cell-wise accuracy of clustering, k=3, N=100, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7781
4	0.8781	0.8443	0.9768	0.9775	0.8208
5	0.8741	0.8427	0.9666	0.9625	0.8287
6	0.8820	0.8463	0.9649	0.9622	0.8386
7	0.8972	0.8517	0.9670	0.9650	0.8523
8	0.9130	0.8581	0.9734	0.9756	0.8640
9	0.9206	0.8626	0.9749	0.9762	0.8788
10	0.9306	0.8704	0.9762	0.9796	0.8906

Table A.8 Vector-wise accuracy of clustering, k=3, N=100, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.4685
4	0.6886	0.5898	0.9357	0.9369	0.4997
5	0.6412	0.5411	0.9019	0.8865	0.4917
6	0.6380	0.5247	0.8906	0.8772	0.4920
7	0.6672	0.5327	0.8903	0.8756	0.5148
8	0.7021	0.5347	0.9025	0.9037	0.5433
9	0.7139	0.5451	0.8981	0.8983	0.5778
10	0.7470	0.5657	0.8991	0.9079	0.6071

A.1.3 N=100, slip=guess=0.2

Table A.9 RMSE of clustering, k=3, N=100, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	0.0000	0.0000	0.0000	0.0000	0.1541
4	0.1393	0.1534	0.0964	0.0936	0.1488
5	0.1496	0.1669	0.1251	0.1303	0.1717
6	0.1508	0.1649	0.1373	0.1569	0.1760
7	0.1507	0.1633	0.1425	0.1628	0.1720
8	0.1427	0.1585	0.1439	0.1639	0.1639
9	0.1404	0.1542	0.1448	0.1588	0.1594
10	0.1359	0.1530	0.1467	0.1572	0.1567

Table A.10 F-score of clustering, k=3, N=100, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7761
4	0.8259	0.8132	0.8976	0.9005	0.7780
5	0.7958	0.7715	0.8584	0.8536	0.7593
6	0.7826	0.7612	0.8367	0.8171	0.7503
7	0.7774	0.7559	0.8289	0.8081	0.7506
8	0.7847	0.7585	0.8250	0.8043	0.7588
9	0.7857	0.7610	0.8201	0.8059	0.7602
10	0.7922	0.7599	0.8182	0.8076	0.7621

Table A.11 Cell-wise accuracy of clustering, k=3, N=100, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7773
4	0.8377	0.8250	0.9036	0.9064	0.7909
5	0.8202	0.7971	0.8749	0.8697	0.7811
6	0.8161	0.7953	0.8627	0.8431	0.7808
7	0.8139	0.7936	0.8575	0.8372	0.7838
8	0.8231	0.7993	0.8561	0.8361	0.7941
9	0.8270	0.8051	0.8552	0.8412	0.7989
10	0.8330	0.8051	0.8533	0.8428	0.8016

Table A.12 Vector-wise accuracy of clustering, N=100, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.4697
4	0.5585	0.5213	0.7231	0.7313	0.4061
5	0.4552	0.3909	0.6130	0.5897	0.3348
6	0.3948	0.3308	0.5359	0.4662	0.2859
7	0.3565	0.2959	0.4808	0.4053	0.2629
8	0.3455	0.2823	0.4312	0.3666	0.2615
9	0.3365	0.2771	0.4047	0.3540	0.2511
10	0.3299	0.2571	0.3681	0.3348	0.2411

A.1.4 N=500, slip=guess=0.05

Table A.13 RMSE of clustering, k=3, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	0.0000	0.0000	0.0000	0.0000	0.1477
4	0.1078	0.1219	0.0000	0.0000	0.1218
5	0.1052	0.1176	0.0000	0.0000	0.1133
6	0.0961	0.1112	0.0000	0.0000	0.1110
7	0.0809	0.0989	0.0000	0.0000	0.0975
8	0.0672	0.0959	0.0000	0.0000	0.0838
9	0.0548	0.0932	0.0000	0.0000	0.0784
10	0.0450	0.0824	0.0000	0.0000	0.0645

Table A.14 F-score of clustering, k=3, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7918
4	0.8651	0.8523	1.0000	1.0000	0.8225
5	0.8555	0.8467	1.0000	1.0000	0.8418
6	0.8633	0.8485	1.0000	1.0000	0.8455
7	0.8813	0.8599	1.0000	1.0000	0.8614
8	0.9022	0.8634	1.0000	1.0000	0.8801
9	0.9194	0.8688	1.0000	1.0000	0.8877
10	0.9346	0.8823	1.0000	1.0000	0.9072

Table A.15 Cell-wise accuracy of clustering, k=3, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7929
4	0.8737	0.8616	1.0000	1.0000	0.8347
5	0.8712	0.8614	1.0000	1.0000	0.8560
6	0.8824	0.8684	1.0000	1.0000	0.8639
7	0.9021	0.8827	1.0000	1.0000	0.8828
8	0.9200	0.8866	1.0000	1.0000	0.9008
9	0.9353	0.8914	1.0000	1.0000	0.9079
10	0.9484	0.9048	1.0000	1.0000	0.9256

Table A.16 Vector-wise accuracy of clustering, k=3, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.5023
4	0.6689	0.6372	1.0000	1.0000	0.5419
5	0.6342	0.5905	1.0000	1.0000	0.5689
6	0.6411	0.5886	1.0000	1.0000	0.5606
7	0.6795	0.6125	1.0000	1.0000	0.5990
8	0.7196	0.6196	1.0000	1.0000	0.6521
9	0.7630	0.6244	1.0000	1.0000	0.6677
10	0.8050	0.6681	1.0000	1.0000	0.7235

A.1.5 N=500, slip=guess=0.1

Table A.17 RMSE of clustering, k=3, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	0.0000	0.0000	0.0000	0.0000	0.1466
4	0.1100	0.1287	0.0000	0.0000	0.1258
5	0.1149	0.1304	0.0001	0.0005	0.1184
6	0.1054	0.1220	0.0000	0.0001	0.1239
7	0.0908	0.1109	0.0000	0.0000	0.1146
8	0.0752	0.1035	0.0000	0.0001	0.0972
9	0.0616	0.0944	0.0000	0.0000	0.0785
10	0.0473	0.0851	0.0000	0.0000	0.0742

Table A.18 F-score of clustering, k=3, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7938
4	0.8667	0.8468	1.0000	1.0000	0.8169
5	0.8436	0.8239	0.9999	0.9995	0.8294
6	0.8522	0.8322	1.0000	0.9999	0.8287
7	0.8669	0.8384	1.0000	1.0000	0.8358
8	0.8911	0.8493	1.0000	0.9999	0.8640
9	0.9128	0.8600	1.0000	1.0000	0.8904
10	0.9374	0.8765	1.0000	1.0000	0.8980

Table A.19 Cell-wise accuracy of clustering, k=3, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7947
4	0.8745	0.8547	1.0000	1.0000	0.8282
5	0.8624	0.8435	0.9999	0.9996	0.8474
6	0.8716	0.8518	1.0000	0.9999	0.8465
7	0.8905	0.8646	1.0000	1.0000	0.8598
8	0.9112	0.8748	1.0000	0.9999	0.8857
9	0.9308	0.8857	1.0000	1.0000	0.9105
10	0.9502	0.8986	1.0000	1.0000	0.9166

Table A.20 Vector-wise accuracy of clustering, k=3, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.5128
4	0.6720	0.6145	1.0000	1.0000	0.5207
5	0.6077	0.5458	0.9997	0.9986	0.5480
6	0.6137	0.5358	1.0000	0.9996	0.5155
7	0.6413	0.5511	1.0000	0.9999	0.5348
8	0.6967	0.5743	0.9998	0.9993	0.6151
9	0.7431	0.5961	1.0000	0.9999	0.6821
10	0.8051	0.6399	1.0000	0.9997	0.7053

A.1.6 N=500, slip=guess=0.2

Table A.21 RMSE of clustering, k=3, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	0.0000	0.0000	0.0000	0.0000	0.1474
4	0.1367	0.1431	0.0127	0.0125	0.1403
5	0.1502	0.1562	0.0313	0.0588	0.1399
6	0.1405	0.1545	0.0246	0.0569	0.1544
7	0.1248	0.1454	0.0234	0.0600	0.1453
8	0.1087	0.1348	0.0191	0.0477	0.1344
9	0.0975	0.1260	0.0174	0.0442	0.1252
10	0.0867	0.1223	0.0121	0.0332	0.1204

Table A.22 F-score of clustering, k=3, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7915
4	0.8358	0.8330	0.9854	0.9856	0.7902
5	0.8013	0.7950	0.9654	0.9352	0.7962
6	0.8015	0.7805	0.9714	0.9344	0.7839
7	0.8227	0.7861	0.9713	0.9278	0.7971
8	0.8508	0.8026	0.9763	0.9413	0.8158
9	0.8722	0.8167	0.9778	0.9440	0.8310
10	0.8948	0.8240	0.9846	0.9576	0.8402

Table A.23 Cell-wise accuracy of clustering, k=3, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.7923
4	0.8469	0.8442	0.9873	0.9875	0.8061
5	0.8222	0.8147	0.9687	0.9412	0.8154
6	0.8304	0.8094	0.9754	0.9431	0.8081
7	0.8536	0.8199	0.9766	0.9400	0.8239
8	0.8780	0.8352	0.9809	0.9523	0.8427
9	0.8980	0.8494	0.9826	0.9558	0.8584
10	0.9158	0.8548	0.9879	0.9668	0.8658

Table A.24 Vector-wise accuracy of clustering, k=3, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
3	1.0000	1.0000	1.0000	1.0000	0.5069
4	0.5809	0.5790	0.9602	0.9605	0.4728
5	0.4785	0.4452	0.8972	0.8030	0.4496
6	0.4782	0.4016	0.9099	0.7882	0.3976
7	0.5218	0.4130	0.8989	0.7541	0.4079
8	0.5800	0.4448	0.8984	0.7695	0.4624
9	0.6284	0.4699	0.8922	0.7615	0.4989
10	0.6801	0.4761	0.9135	0.8062	0.5045

A.2 Results of Clustering for IRPtoQ: 4-skill case

A.2.1 N=200, slip=guess=0.05

Table A.25 RMSE of clustering, k=4, N=200, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	0.0000	0.0000	0.0000	0.0000	0.1596
5	0.1147	0.1381	0.0050	0.0048	0.1268
6	0.1258	0.1469	0.0076	0.0084	0.1238
7	0.1181	0.1411	0.0088	0.0091	0.1335
8	0.1104	0.1347	0.0089	0.0103	0.1321
9	0.1038	0.1285	0.0093	0.0098	0.1270
10	0.0960	0.1253	0.0096	0.0091	0.1200
11	0.0938	0.1261	0.0101	0.0078	0.1209
12	0.0834	0.1137	0.0081	0.0049	0.1156
13	0.0868	0.1193	0.0085	0.0052	0.1116
14	0.0811	0.1170	0.0122	0.0096	0.1032
15	0.0728	0.1064	0.0056	0.0062	0.1151
16	0.0719	0.1116	0.0082	0.0059	0.1053
17	0.0722	0.1084	0.0062	0.0057	0.0984
18	0.0669	0.1035	0.0070	0.0057	0.1043
19	0.0720	0.1155	0.0040	0.0013	0.1081
20	0.0691	0.0988	0.0029	0.0048	0.1012

Table A.26 F-score of clustering, k=4, N=200, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7727
5	0.8574	0.8360	0.9946	0.9948	0.8151
6	0.8323	0.8106	0.9913	0.9904	0.8268
7	0.8345	0.8086	0.9893	0.9888	0.8161
8	0.8403	0.8097	0.9886	0.9869	0.8166
9	0.8451	0.8134	0.9878	0.9873	0.8193
10	0.8536	0.8150	0.9866	0.9874	0.8257
11	0.8575	0.8154	0.9862	0.9892	0.8256
12	0.8708	0.8262	0.9883	0.9929	0.8282
13	0.8676	0.8233	0.9878	0.9924	0.8351
14	0.8720	0.8193	0.9820	0.9857	0.8430
15	0.8801	0.8251	0.9905	0.9895	0.8153
16	0.8889	0.8268	0.9879	0.9913	0.8399
17	0.8862	0.8282	0.9907	0.9911	0.8445
18	0.8926	0.8349	0.9892	0.9912	0.8357
19	0.8845	0.8173	0.9937	0.9979	0.8326
20	0.8960	0.8478	0.9957	0.9926	0.8476

Table A.27 Cell-wise accuracy of clustering, k=4, N=200, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7738
5	0.8683	0.8475	0.9950	0.9952	0.8315
6	0.8545	0.8337	0.9924	0.9916	0.8476
7	0.8639	0.8404	0.9912	0.9909	0.8452
8	0.8740	0.8476	0.9911	0.9897	0.8508
9	0.8818	0.8549	0.9907	0.9902	0.8572
10	0.8916	0.8606	0.9904	0.9909	0.8655
11	0.8943	0.8599	0.9899	0.9922	0.8662
12	0.9084	0.8742	0.9919	0.9951	0.8732
13	0.9050	0.8693	0.9915	0.9948	0.8775
14	0.9120	0.8720	0.9878	0.9904	0.8875
15	0.9233	0.8842	0.9944	0.9938	0.8759
16	0.9242	0.8786	0.9918	0.9941	0.8861
17	0.9239	0.8820	0.9938	0.9943	0.8931
18	0.9297	0.8884	0.9930	0.9943	0.8871
19	0.9240	0.8755	0.9960	0.9987	0.8849
20	0.9284	0.8938	0.9971	0.9952	0.8913

Table A.28 Vector-wise accuracy of clustering, k=4, N=200, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.3426
5	0.5960	0.5225	0.9833	0.9839	0.4287
6	0.5200	0.4366	0.9754	0.9719	0.4526
7	0.5216	0.4277	0.9712	0.9697	0.4321
8	0.5412	0.4345	0.9700	0.9635	0.4349
9	0.5476	0.4368	0.9667	0.9636	0.4397
10	0.5672	0.4531	0.9684	0.9664	0.4473
11	0.5777	0.4434	0.9598	0.9660	0.4461
12	0.6086	0.4840	0.9707	0.9770	0.4730
13	0.6148	0.4711	0.9652	0.9754	0.4875
14	0.6324	0.4816	0.9520	0.9590	0.5238
15	0.6472	0.4826	0.9667	0.9688	0.4743
16	0.6582	0.4906	0.9621	0.9723	0.5137
17	0.6836	0.5066	0.9688	0.9707	0.5367
18	0.6754	0.5039	0.9637	0.9730	0.4996
19	0.6535	0.4826	0.9750	0.9847	0.4979
20	0.7097	0.5604	0.9785	0.9708	0.5618

A.2.2 N=200, slip=guess=0.1

Table A.29 RMSE of clustering, k=4, N=200, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	0.0000	0.0000	0.0000	0.0000	0.1597
5	0.1333	0.1513	0.0299	0.0295	0.1340
6	0.1425	0.1558	0.0430	0.0464	0.1444
7	0.1343	0.1504	0.0492	0.0559	0.1561
8	0.1292	0.1458	0.0513	0.0585	0.1543
9	0.1210	0.1399	0.0537	0.0601	0.1465
10	0.1158	0.1364	0.0545	0.0607	0.1417
11	0.1101	0.1318	0.0537	0.0586	0.1365
12	0.1020	0.1283	0.0517	0.0578	0.1301
13	0.1009	0.1261	0.0508	0.0579	0.1292
14	0.0943	0.1205	0.0484	0.0547	0.1236
15	0.0891	0.1189	0.0493	0.0563	0.1216
16	0.0883	0.1191	0.0494	0.0553	0.1191
17	0.0856	0.1159	0.0500	0.0554	0.1192
18	0.0826	0.1155	0.0522	0.0568	0.1163
19	0.0793	0.1146	0.0511	0.0570	0.1149
20	0.0781	0.1137	0.0524	0.0605	0.1159

Table A.30 F-score of clustering, k=4, N=200, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7729
5	0.8356	0.8189	0.9678	0.9682	0.8018
6	0.8109	0.7966	0.9505	0.9466	0.7987
7	0.8095	0.7899	0.9395	0.9319	0.7818
8	0.8083	0.7857	0.9334	0.9249	0.7771
9	0.8172	0.7907	0.9285	0.9204	0.7844
10	0.8215	0.7908	0.9258	0.9181	0.7861
11	0.8269	0.7915	0.9246	0.9188	0.7901
12	0.8414	0.7972	0.9273	0.9190	0.7996
13	0.8369	0.7932	0.9251	0.9152	0.7944
14	0.8483	0.8019	0.9281	0.9191	0.8024
15	0.8565	0.8033	0.9252	0.9151	0.8038
16	0.8586	0.8038	0.9257	0.9172	0.8087
17	0.8632	0.8090	0.9244	0.9166	0.8079
18	0.8677	0.8089	0.9206	0.9134	0.8111
19	0.8737	0.8102	0.9217	0.9127	0.8126
20	0.8720	0.8080	0.9174	0.9047	0.8074

Table A.31 Cell-wise accuracy of clustering, k=4, N=200, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7736
5	0.8473	0.8310	0.9701	0.9705	0.8177
6	0.8339	0.8202	0.9570	0.9536	0.8199
7	0.8432	0.8250	0.9508	0.9441	0.8142
8	0.8496	0.8291	0.9487	0.9415	0.8183
9	0.8589	0.8358	0.9463	0.9399	0.8272
10	0.8662	0.8397	0.9455	0.9393	0.8325
11	0.8737	0.8453	0.9463	0.9414	0.8397
12	0.8848	0.8503	0.9483	0.9422	0.8481
13	0.8865	0.8530	0.9492	0.9421	0.8496
14	0.8954	0.8605	0.9516	0.9453	0.8572
15	0.9031	0.8635	0.9507	0.9437	0.8604
16	0.9041	0.8635	0.9506	0.9447	0.8635
17	0.9079	0.8680	0.9500	0.9446	0.8638
18	0.9121	0.8693	0.9478	0.9432	0.8678
19	0.9167	0.8707	0.9489	0.9430	0.8696
20	0.9182	0.8724	0.9476	0.9395	0.8690

Table A.32 Vector-wise accuracy of clustering, k=4, N=200, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.3463
5	0.5262	0.4716	0.9008	0.9029	0.3889
6	0.4470	0.3986	0.8597	0.8473	0.3687
7	0.4427	0.3756	0.8330	0.8059	0.3266
8	0.4365	0.3588	0.8195	0.7851	0.3127
9	0.4492	0.3590	0.8012	0.7706	0.3200
10	0.4627	0.3579	0.7883	0.7537	0.3286
11	0.4768	0.3690	0.7828	0.7526	0.3395
12	0.5038	0.3822	0.7750	0.7484	0.3570
13	0.5017	0.3753	0.7610	0.7347	0.3490
14	0.5345	0.3974	0.7583	0.7368	0.3762
15	0.5471	0.4027	0.7425	0.7208	0.3801
16	0.5573	0.4076	0.7247	0.7078	0.3954
17	0.5724	0.4201	0.7119	0.6931	0.3940
18	0.5844	0.4174	0.6842	0.6697	0.4003
19	0.5973	0.4261	0.6744	0.6558	0.4077
20	0.5933	0.4271	0.6421	0.6184	0.3920

A.2.3 N=200, slip=guess=0.2

Table A.33 RMSE of clustering, k=4, N=200, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	0.0000	0.0000	0.0000	0.0000	0.1593
5	0.1530	0.1612	0.0921	0.0918	0.1512
6	0.1695	0.1758	0.1243	0.1395	0.1724
7	0.1708	0.1765	0.1465	0.1718	0.1877
8	0.1669	0.1743	0.1589	0.1848	0.1845
9	0.1629	0.1717	0.1640	0.1859	0.1812
10	0.1603	0.1669	0.1709	0.1895	0.1761
11	0.1581	0.1666	0.1741	0.1897	0.1744
12	0.1553	0.1637	0.1738	0.1877	0.1708
13	0.1522	0.1607	0.1762	0.1867	0.1671
14	0.1499	0.1595	0.1760	0.1836	0.1651
15	0.1475	0.1567	0.1738	0.1828	0.1637
16	0.1450	0.1557	0.1734	0.1800	0.1611
17	0.1438	0.1536	0.1752	0.1804	0.1604
18	0.1401	0.1515	0.1730	0.1793	0.1580
19	0.1386	0.1505	0.1742	0.1801	0.1557
20	0.1371	0.1511	0.1791	0.1817	0.1537

Table A.34 F-score of clustering, k=4, N=200, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7723
5	0.8119	0.8048	0.9004	0.9008	0.7729
6	0.7698	0.7620	0.8549	0.8384	0.7530
7	0.7514	0.7441	0.8199	0.7935	0.7296
8	0.7448	0.7323	0.7986	0.7714	0.7235
9	0.7388	0.7242	0.7838	0.7610	0.7153
10	0.7383	0.7266	0.7746	0.7554	0.7192
11	0.7350	0.7190	0.7658	0.7500	0.7146
12	0.7352	0.7200	0.7637	0.7496	0.7166
13	0.7316	0.7149	0.7531	0.7430	0.7126
14	0.7339	0.7151	0.7526	0.7457	0.7146
15	0.7362	0.7174	0.7547	0.7453	0.7143
16	0.7403	0.7179	0.7537	0.7478	0.7173
17	0.7381	0.7166	0.7476	0.7431	0.7137
18	0.7437	0.7168	0.7480	0.7412	0.7151
19	0.7481	0.7212	0.7479	0.7419	0.7215
20	0.7601	0.7273	0.7478	0.7463	0.7333

Table A.35 Cell-wise accuracy of clustering, k=4, N=200, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7730
5	0.8256	0.8182	0.9079	0.9082	0.7895
6	0.7990	0.7905	0.8757	0.8605	0.7787
7	0.7935	0.7847	0.8535	0.8282	0.7670
8	0.7954	0.7828	0.8411	0.8152	0.7697
9	0.7989	0.7846	0.8360	0.8141	0.7711
10	0.8000	0.7882	0.8291	0.8105	0.7760
11	0.8018	0.7873	0.8259	0.8103	0.7767
12	0.8046	0.7906	0.8262	0.8123	0.7812
13	0.8085	0.7934	0.8238	0.8133	0.7846
14	0.8116	0.7951	0.8240	0.8164	0.7873
15	0.8144	0.7982	0.8262	0.8172	0.7886
16	0.8184	0.7998	0.8266	0.8200	0.7921
17	0.8200	0.8025	0.8248	0.8196	0.7929
18	0.8263	0.8052	0.8270	0.8207	0.7963
19	0.8284	0.8072	0.8258	0.8199	0.8001
20	0.8322	0.8067	0.8209	0.8183	0.8038

Table A.36 Vector-wise accuracy of clustering, k=4, N=200, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.00000	1.00000	1.00000	1.00000	0.33892
5	0.44067	0.41396	0.69336	0.69561	0.30366
6	0.30776	0.27827	0.56196	0.50781	0.22925
7	0.24780	0.21270	0.44155	0.34590	0.17017
8	0.21616	0.17856	0.35894	0.26074	0.15371
9	0.19189	0.15854	0.29409	0.22036	0.13335
10	0.18032	0.14541	0.24517	0.18950	0.13047
11	0.16709	0.13291	0.21074	0.16733	0.11611
12	0.16216	0.12686	0.18672	0.15454	0.11128
13	0.15166	0.11841	0.15786	0.13882	0.10337
14	0.14731	0.11582	0.14199	0.12939	0.10791
15	0.14790	0.11294	0.13037	0.11680	0.10093
16	0.15195	0.11040	0.11255	0.10327	0.10264
17	0.14429	0.10933	0.09624	0.08774	0.09580
18	0.15137	0.10908	0.08389	0.07812	0.09531
19	0.15039	0.10498	0.07124	0.06753	0.09897
20	0.15057	0.10227	0.06023	0.05455	0.09716

A.2.4 N=500, slip=guess=0.05

Table A.37 RMSE of clustering, k=4, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	0.0000	0.0000	0.0000	0.0000	0.1572
5	0.1312	0.1432	0.0000	0.0000	0.1237
6	0.1370	0.1448	0.0001	0.0001	0.1256
7	0.1275	0.1353	0.0000	0.0001	0.1341
8	0.1161	0.1285	0.0000	0.0000	0.1281
9	0.1049	0.1240	0.0000	0.0000	0.1213
10	0.0984	0.1201	0.0000	0.0000	0.1144
11	0.0888	0.1113	0.0000	0.0000	0.1096
12	0.0863	0.1128	0.0000	0.0000	0.1056
13	0.0782	0.1077	0.0000	0.0000	0.1048
14	0.0749	0.1045	0.0000	0.0000	0.0967
15	0.0712	0.1021	0.0000	0.0001	0.0948
16	0.0697	0.1016	0.0000	0.0000	0.0969
17	0.0668	0.1005	0.0000	0.0000	0.0926
18	0.0641	0.0984	0.0000	0.0000	0.0916
19	0.0621	0.0943	0.0000	0.0000	0.0906
20	0.0608	0.0926	0.0000	0.0000	0.0884

Table A.38 F-score of clustering, k=4, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	0.0000	0.0000	0.0000	0.0000	0.1572
5	0.1312	0.1432	0.0000	0.0000	0.1237
6	0.1370	0.1448	0.0001	0.0001	0.1256
7	0.1275	0.1353	0.0000	0.0001	0.1341
8	0.1161	0.1285	0.0000	0.0000	0.1281
9	0.1049	0.1240	0.0000	0.0000	0.1213
10	0.0984	0.1201	0.0000	0.0000	0.1144
11	0.0888	0.1113	0.0000	0.0000	0.1096
12	0.0863	0.1128	0.0000	0.0000	0.1056
13	0.0782	0.1077	0.0000	0.0000	0.1048
14	0.0749	0.1045	0.0000	0.0000	0.0967
15	0.0712	0.1021	0.0000	0.0001	0.0948
16	0.0697	0.1016	0.0000	0.0000	0.0969
17	0.0668	0.1005	0.0000	0.0000	0.0926
18	0.0641	0.0984	0.0000	0.0000	0.0916
19	0.0621	0.0943	0.0000	0.0000	0.0906
20	0.0608	0.0926	0.0000	0.0000	0.0884

Table A.39 Cell-wise accuracy of clustering, k=4, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7760
5	0.8474	0.8385	1.0000	1.0000	0.8338
6	0.8382	0.8325	0.9999	0.9999	0.8432
7	0.8490	0.8420	1.0000	0.9999	0.8392
8	0.8629	0.8497	1.0000	1.0000	0.8484
9	0.8769	0.8557	1.0000	1.0000	0.8578
10	0.8854	0.8614	1.0000	1.0000	0.8670
11	0.8970	0.8714	1.0000	1.0000	0.8706
12	0.9010	0.8709	1.0000	1.0000	0.8802
13	0.9111	0.8769	1.0000	1.0000	0.8802
14	0.9155	0.8815	1.0000	1.0000	0.8901
15	0.9203	0.8848	1.0000	0.9999	0.8930
16	0.9226	0.8860	1.0000	1.0000	0.8912
17	0.9262	0.8872	1.0000	1.0000	0.8966
18	0.9294	0.8902	1.0000	1.0000	0.8981
19	0.9321	0.8949	1.0000	1.0000	0.8998
20	0.9337	0.8976	1.0000	1.0000	0.9024

Table A.40 Vector-wise accuracy of clustering, k=4, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.3528
5	0.5268	0.4940	1.0000	0.9999	0.4451
6	0.4600	0.4329	0.9998	0.9998	0.4394
7	0.4598	0.4276	1.0000	0.9998	0.3992
8	0.4899	0.4278	1.0000	1.0000	0.4111
9	0.5207	0.4368	1.0000	1.0000	0.4271
10	0.5415	0.4504	0.9999	1.0000	0.4510
11	0.5528	0.4597	1.0000	1.0000	0.4451
12	0.5891	0.4722	1.0000	1.0000	0.4935
13	0.6104	0.4827	1.0000	1.0000	0.4853
14	0.6312	0.4963	1.0000	1.0000	0.5206
15	0.6504	0.5065	1.0000	0.9998	0.5366
16	0.6501	0.5071	0.9999	1.0000	0.5230
17	0.6678	0.5194	1.0000	1.0000	0.5516
18	0.6777	0.5295	1.0000	1.0000	0.5524
19	0.6929	0.5474	1.0000	1.0000	0.5694
20	0.7083	0.5636	0.9999	0.9999	0.5809

A.2.5 N=500, slip=guess=0.1

Table A.41 RMSE of clustering, k=4, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	0.0000	0.0000	0.0000	0.0000	0.1565
5	0.1421	0.1483	0.0016	0.0015	0.1304
6	0.1470	0.1532	0.0045	0.0074	0.1399
7	0.1381	0.1461	0.0059	0.0099	0.1514
8	0.1281	0.1414	0.0049	0.0087	0.1447
9	0.1160	0.1334	0.0052	0.0084	0.1347
10	0.1062	0.1259	0.0047	0.0066	0.1256
11	0.0980	0.1195	0.0043	0.0060	0.1208
12	0.0918	0.1158	0.0048	0.0064	0.1157
13	0.0837	0.1113	0.0044	0.0051	0.1079
14	0.0810	0.1088	0.0035	0.0041	0.1034
15	0.0744	0.1039	0.0031	0.0036	0.1034
16	0.0702	0.1016	0.0034	0.0029	0.0998
17	0.0672	0.0979	0.0024	0.0024	0.0963
18	0.0621	0.0973	0.0025	0.0025	0.0954
19	0.0624	0.0961	0.0029	0.0025	0.0953
20	0.0600	0.0977	0.0036	0.0032	0.0959

Table A.42 F-score of clustering, k=4, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7768
5	0.8260	0.8219	0.9982	0.9983	0.8071
6	0.8009	0.7964	0.9947	0.9915	0.7973
7	0.8014	0.7941	0.9929	0.9883	0.7821
8	0.8064	0.7905	0.9937	0.9889	0.7860
9	0.8204	0.7965	0.9933	0.9892	0.7976
10	0.8323	0.8026	0.9936	0.9911	0.8065
11	0.8444	0.8105	0.9941	0.9917	0.8138
12	0.8497	0.8095	0.9931	0.9908	0.8150
13	0.8661	0.8198	0.9936	0.9927	0.8309
14	0.8692	0.8191	0.9947	0.9938	0.8351
15	0.8796	0.8260	0.9953	0.9945	0.8340
16	0.8865	0.8289	0.9947	0.9954	0.8384
17	0.8912	0.8354	0.9962	0.9962	0.8437
18	0.8989	0.8323	0.9961	0.9960	0.8419
19	0.8998	0.8361	0.9954	0.9959	0.8432
20	0.9031	0.8312	0.9941	0.9948	0.8406

Table A.43 Cell-wise accuracy of clustering, k=4, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7771
5	0.8373	0.8332	0.9984	0.9985	0.8227
6	0.8262	0.8212	0.9955	0.9926	0.8212
7	0.8347	0.8269	0.9941	0.9901	0.8135
8	0.8466	0.8316	0.9951	0.9913	0.8237
9	0.8617	0.8404	0.9948	0.9916	0.8373
10	0.8745	0.8495	0.9953	0.9934	0.8487
11	0.8858	0.8576	0.9957	0.9940	0.8568
12	0.8940	0.8623	0.9952	0.9936	0.8635
13	0.9048	0.8686	0.9956	0.9949	0.8744
14	0.9093	0.8712	0.9965	0.9959	0.8804
15	0.9182	0.8784	0.9969	0.9964	0.8813
16	0.9237	0.8815	0.9966	0.9971	0.8865
17	0.9274	0.8868	0.9976	0.9976	0.8907
18	0.9344	0.8875	0.9975	0.9975	0.8923
19	0.9345	0.8897	0.9971	0.9975	0.8927
20	0.9377	0.8877	0.9964	0.9968	0.8921

Table A.44 Vector-wise accuracy of clustering, k=4, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.3552
5	0.4884	0.4760	0.9941	0.9942	0.4082
6	0.4128	0.3962	0.9855	0.9757	0.3814
7	0.4069	0.3729	0.9804	0.9659	0.3226
8	0.4252	0.3618	0.9839	0.9676	0.3268
9	0.4606	0.3736	0.9812	0.9672	0.3531
10	0.4889	0.3869	0.9815	0.9707	0.3800
11	0.5192	0.4048	0.9816	0.9737	0.4031
12	0.5415	0.4111	0.9776	0.9687	0.4217
13	0.5856	0.4463	0.9793	0.9742	0.4659
14	0.6017	0.4473	0.9811	0.9768	0.4761
15	0.6200	0.4642	0.9828	0.9795	0.4800
16	0.6433	0.4815	0.9804	0.9803	0.5063
17	0.6587	0.4933	0.9818	0.9820	0.5180
18	0.6789	0.5006	0.9784	0.9775	0.5248
19	0.6870	0.5051	0.9759	0.9760	0.5240
20	0.6999	0.5020	0.9671	0.9700	0.5208

A.2.6 N=500, slip=guess=0.2

Table A.45 RMSE of clustering, k=4, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	0.0000	0.0000	0.0000	0.0000	0.1560
5	0.1554	0.1613	0.0411	0.0405	0.1463
6	0.1688	0.1703	0.0709	0.0956	0.1584
7	0.1652	0.1732	0.0795	0.1321	0.1789
8	0.1576	0.1669	0.0906	0.1494	0.1748
9	0.1490	0.1610	0.1022	0.1547	0.1688
10	0.1420	0.1552	0.1067	0.1525	0.1610
11	0.1393	0.1521	0.1115	0.1531	0.1579
12	0.1335	0.1484	0.1151	0.1448	0.1532
13	0.1276	0.1446	0.1119	0.1372	0.1488
14	0.1259	0.1429	0.1151	0.1358	0.1475
15	0.1220	0.1403	0.1137	0.1320	0.1436
16	0.1179	0.1369	0.1116	0.1268	0.1401
17	0.1161	0.1352	0.1155	0.1272	0.1378
18	0.1119	0.1317	0.1134	0.1232	0.1352
19	0.1111	0.1299	0.1145	0.1247	0.1340
20	0.1059	0.1268	0.1139	0.1212	0.1303

Table A.46 F-score of clustering, k=4, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7780
5	0.8108	0.8073	0.9554	0.9560	0.7764
6	0.7710	0.7710	0.9178	0.8900	0.7627
7	0.7554	0.7462	0.9006	0.8381	0.7342
8	0.7556	0.7434	0.8821	0.8115	0.7321
9	0.7660	0.7471	0.8658	0.8023	0.7377
10	0.7748	0.7529	0.8590	0.8036	0.7478
11	0.7689	0.7463	0.8446	0.7921	0.7421
12	0.7772	0.7504	0.8377	0.8004	0.7481
13	0.7868	0.7528	0.8399	0.8078	0.7530
14	0.7856	0.7516	0.8324	0.8059	0.7517
15	0.7931	0.7547	0.8333	0.8095	0.7585
16	0.8001	0.7598	0.8346	0.8145	0.7633
17	0.8067	0.7643	0.8306	0.8154	0.7695
18	0.8158	0.7703	0.8324	0.8196	0.7733
19	0.8145	0.7704	0.8273	0.8136	0.7723
20	0.8274	0.7786	0.8287	0.8193	0.7810

Table A.47 Cell-wise accuracy of clustering, k=4, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.7788
5	0.8242	0.8202	0.9589	0.9595	0.7934
6	0.7980	0.7970	0.9291	0.9044	0.7879
7	0.7984	0.7883	0.9205	0.8679	0.7739
8	0.8059	0.7922	0.9094	0.8506	0.7784
9	0.8167	0.7983	0.8978	0.8453	0.7853
10	0.8257	0.8047	0.8933	0.8475	0.7958
11	0.8303	0.8093	0.8885	0.8469	0.8008
12	0.8389	0.8149	0.8849	0.8552	0.8078
13	0.8481	0.8195	0.8881	0.8628	0.8143
14	0.8507	0.8220	0.8849	0.8642	0.8169
15	0.8571	0.8254	0.8863	0.8680	0.8234
16	0.8638	0.8310	0.8884	0.8732	0.8288
17	0.8674	0.8333	0.8845	0.8728	0.8334
18	0.8747	0.8391	0.8866	0.8768	0.8370
19	0.8768	0.8423	0.8855	0.8753	0.8398
20	0.8853	0.8479	0.8861	0.8788	0.8462

Table A.48 Vector-wise accuracy of clustering, k=4, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB
4	1.0000	1.0000	1.0000	1.0000	0.3598
5	0.4339	0.4259	0.8608	0.8637	0.3197
6	0.3063	0.3012	0.7588	0.6653	0.2698
7	0.2671	0.2323	0.6991	0.5024	0.2026
8	0.2583	0.2092	0.6380	0.4126	0.1813
9	0.2697	0.2062	0.5651	0.3583	0.1878
10	0.2799	0.2132	0.5201	0.3399	0.1907
11	0.2778	0.2095	0.4702	0.3068	0.1889
12	0.2921	0.2130	0.4399	0.3202	0.1955
13	0.3082	0.2178	0.4172	0.3200	0.1988
14	0.3089	0.2192	0.3840	0.3041	0.2060
15	0.3252	0.2269	0.3572	0.2964	0.2151
16	0.3411	0.2386	0.3364	0.2854	0.2236
17	0.3540	0.2428	0.2949	0.2626	0.2334
18	0.3651	0.2553	0.2627	0.2361	0.2375
19	0.3761	0.2591	0.2259	0.2040	0.2432
20	0.4012	0.2738	0.1987	0.1837	0.2571

APPENDIX B EXPERIMENT RESULTS OF Q-MATRIX DERIVATION

B.1 Results of Q-matrix Derivation: 3-skill case

B.1.1 N=100, slip=guess=0.05

Table B.1 F-score of derivation, k=3, N=100, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8590	0.9924	0.3356	0.9984	0.7499
4	0.9231	0.8757	0.9975	0.9973	0.8644	0.9730	0.5989	0.9825	0.7740
5	0.8896	0.8314	0.9678	0.9679	0.8519	0.9429	0.6854	0.9744	0.7819
6	0.8922	0.8359	0.9743	0.9761	0.8528	0.9446	0.7184	0.9760	0.7940
7	0.9015	0.8421	0.9728	0.9731	0.8558	0.9436	0.7292	0.9727	0.8063
8	0.9043	0.8390	0.9698	0.9711	0.8557	0.9360	0.7427	0.9702	0.8091
9	0.9205	0.8526	0.9836	0.9852	0.8713	0.9442	0.7439	0.9762	0.8131
10	0.9212	0.8479	0.9815	0.9830	0.8756	0.9396	0.7450	0.9700	0.8215

Table B.2 Cell-wise accuracy of derivation, k=3, N=100, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8899	0.9940	0.5556	0.9991	0.7644
4	0.9357	0.8971	0.9980	0.9978	0.8805	0.9761	0.6650	0.9850	0.7741
5	0.9029	0.8529	0.9701	0.9702	0.8681	0.9481	0.7281	0.9766	0.7738
6	0.8997	0.8482	0.9746	0.9763	0.8648	0.9464	0.7443	0.9780	0.7851
7	0.9031	0.8466	0.9715	0.9718	0.8607	0.9426	0.7473	0.9740	0.7922
8	0.9045	0.8413	0.9673	0.9685	0.8593	0.9340	0.7602	0.9708	0.7918
9	0.9171	0.8481	0.9819	0.9835	0.8685	0.9395	0.7556	0.9773	0.7971
10	0.9183	0.8438	0.9798	0.9813	0.8730	0.9346	0.7575	0.9704	0.8021

Table B.3 Vector-wise accuracy of derivation, k=3, N=100, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.7180	0.9789	0.3323	0.9974	0.4206
4	0.8668	0.7912	0.9955	0.9955	0.7109	0.9277	0.3670	0.9586	0.4336
5	0.7823	0.6827	0.9120	0.9122	0.7033	0.8422	0.4372	0.9352	0.4194
6	0.7742	0.6712	0.9279	0.9319	0.7079	0.8402	0.4359	0.9401	0.4281
7	0.7801	0.6612	0.9181	0.9185	0.7023	0.8267	0.4326	0.9292	0.4494
8	0.7858	0.6491	0.9067	0.9090	0.6983	0.8034	0.4454	0.9208	0.4447
9	0.8180	0.6692	0.9497	0.9535	0.7247	0.8221	0.4315	0.9383	0.4542
10	0.8204	0.6525	0.9455	0.9486	0.7295	0.8061	0.4252	0.9198	0.4594

B.1.2 N=100, slip=guess=0.1

Table B.4 F-score of derivation, k=3, N=100, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8618	0.9929	0.3354	0.9994	0.7494
4	0.8968	0.8450	0.9787	0.9790	0.8450	0.9628	0.5827	0.9759	0.7666
5	0.8640	0.8258	0.9550	0.9492	0.8165	0.9374	0.6678	0.9644	0.7711
6	0.8602	0.8169	0.9491	0.9459	0.8072	0.9314	0.7109	0.9629	0.7913
7	0.8678	0.8170	0.9482	0.9448	0.8146	0.9255	0.7305	0.9561	0.7926
8	0.8795	0.8213	0.9453	0.9487	0.8216	0.9278	0.7350	0.9563	0.8089
9	0.8885	0.8240	0.9520	0.9510	0.8398	0.9291	0.7386	0.9569	0.8099
10	0.8974	0.8362	0.9503	0.9529	0.8549	0.9362	0.7486	0.9542	0.8157

Table B.5 Cell-wise accuracy of derivation, k=3, N=100, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8928	0.9943	0.5567	0.9997	0.7641
4	0.9145	0.8732	0.9825	0.9828	0.8669	0.9679	0.6640	0.9801	0.7704
5	0.8789	0.8456	0.9594	0.9549	0.8380	0.9423	0.7121	0.9690	0.7657
6	0.8706	0.8314	0.9518	0.9490	0.8250	0.9343	0.7388	0.9661	0.7789
7	0.8729	0.8257	0.9490	0.9457	0.8263	0.9253	0.7507	0.9594	0.7766
8	0.8788	0.8228	0.9430	0.9473	0.8254	0.9255	0.7487	0.9586	0.7915
9	0.8867	0.8228	0.9497	0.9490	0.8404	0.9254	0.7518	0.9585	0.7898
10	0.8929	0.8322	0.9465	0.9494	0.8528	0.9317	0.7601	0.9562	0.7957

Table B.6 Vector-wise accuracy of derivation, k=3, N=100, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.7318	0.9818	0.3344	0.9990	0.4219
4	0.8250	0.7443	0.9625	0.9637	0.6871	0.9064	0.3924	0.9482	0.4236
5	0.7389	0.6745	0.8945	0.8847	0.6530	0.8348	0.4113	0.9206	0.4028
6	0.7204	0.6391	0.8831	0.8768	0.6358	0.8091	0.4341	0.9116	0.4203
7	0.7228	0.6273	0.8751	0.8689	0.6362	0.7901	0.4302	0.8942	0.4138
8	0.7408	0.6209	0.8604	0.8733	0.6368	0.7896	0.4231	0.8928	0.4410
9	0.7552	0.6199	0.8826	0.8847	0.6620	0.7888	0.4194	0.8913	0.4355
10	0.7648	0.6292	0.8694	0.8778	0.6823	0.8045	0.4395	0.8865	0.4478

B.1.3 N=100, slip=guess=0.2

Table B.7 F-score of derivation, k=3, N=100, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8621	0.9940	0.3355	0.9995	0.7488
4	0.8322	0.8142	0.9075	0.9081	0.7970	0.9222	0.5149	0.9235	0.7520
5	0.7858	0.7626	0.8472	0.8334	0.7526	0.8585	0.5899	0.9103	0.7577
6	0.7737	0.7518	0.8210	0.7947	0.7401	0.8679	0.6271	0.8955	0.7678
7	0.7648	0.7409	0.8052	0.7786	0.7361	0.8528	0.6615	0.8853	0.7735
8	0.7681	0.7451	0.7958	0.7756	0.7428	0.8610	0.6796	0.8898	0.7834
9	0.7695	0.7446	0.7981	0.7783	0.7445	0.8592	0.6881	0.8837	0.7916
10	0.7671	0.7391	0.7777	0.7725	0.7413	0.8594	0.7037	0.8881	0.7915

Table B.8 Cell-wise accuracy of derivation, k=3, N=100, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8924	0.9951	0.5568	0.9997	0.7635
4	0.8637	0.8505	0.9253	0.9260	0.8314	0.9356	0.6299	0.9403	0.7534
5	0.8117	0.7927	0.8654	0.8537	0.7844	0.8743	0.6638	0.9272	0.7524
6	0.7896	0.7713	0.8311	0.8099	0.7617	0.8718	0.6728	0.9109	0.7557
7	0.7773	0.7560	0.8135	0.7913	0.7527	0.8534	0.6924	0.8995	0.7556
8	0.7755	0.7542	0.8003	0.7838	0.7538	0.8585	0.7007	0.9010	0.7627
9	0.7720	0.7488	0.7988	0.7815	0.7498	0.8530	0.7040	0.8929	0.7680
10	0.7673	0.7410	0.7764	0.7737	0.7438	0.8514	0.7158	0.8980	0.7704

Table B.9 Vector-wise accuracy of derivation, k=3, N=100, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.7273	0.9833	0.3346	0.9992	0.4172
4	0.7332	0.7080	0.8447	0.8467	0.6393	0.8389	0.3658	0.8605	0.4027
5	0.6173	0.5828	0.7116	0.6867	0.5656	0.7081	0.3767	0.8311	0.3930
6	0.5711	0.5332	0.6422	0.6069	0.5227	0.6892	0.3535	0.7935	0.3875
7	0.5502	0.5044	0.6103	0.5756	0.4992	0.6460	0.3503	0.7691	0.3875
8	0.5403	0.4974	0.5804	0.5591	0.4957	0.6567	0.3628	0.7677	0.3916
9	0.5345	0.4937	0.5850	0.5601	0.4916	0.6436	0.3596	0.7526	0.4010
10	0.5272	0.4738	0.5398	0.5390	0.4755	0.6377	0.3728	0.7668	0.4001

B.1.4 N=500, slip=guess=0.05

Table B.10 F-score of derivation, k=3, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8920	0.9963	0.3333	0.9994	0.7402
4	0.8976	0.8753	1.0000	1.0000	0.8691	0.9749	0.6196	0.9875	0.7682
5	0.8598	0.8429	0.9799	0.9799	0.8577	0.9540	0.6721	0.9881	0.7828
6	0.8565	0.8378	0.9730	0.9730	0.8449	0.9479	0.6955	0.9866	0.7878
7	0.8705	0.8433	0.9740	0.9740	0.8572	0.9532	0.6908	0.9881	0.8010
8	0.8947	0.8550	0.9857	0.9857	0.8818	0.9519	0.7076	0.9849	0.8148
9	0.9039	0.8587	0.9690	0.9690	0.8863	0.9496	0.7087	0.9837	0.8165
10	0.9243	0.8730	0.9808	0.9808	0.9047	0.9562	0.7128	0.9876	0.8238

Table B.11 Cell-wise accuracy of derivation, k=3, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.9152	0.9971	0.5556	0.9997	0.7531
4	0.9142	0.8962	1.0000	1.0000	0.8861	0.9784	0.6829	0.9889	0.7696
5	0.8746	0.8617	0.9812	0.9812	0.8715	0.9583	0.7173	0.9892	0.7764
6	0.8648	0.8480	0.9727	0.9727	0.8537	0.9499	0.7239	0.9875	0.7748
7	0.8717	0.8455	0.9728	0.9728	0.8594	0.9521	0.7108	0.9887	0.7878
8	0.8932	0.8542	0.9840	0.9840	0.8800	0.9500	0.7254	0.9853	0.7990
9	0.9001	0.8556	0.9653	0.9653	0.8828	0.9457	0.7254	0.9837	0.7978
10	0.9196	0.8672	0.9786	0.9786	0.8995	0.9530	0.7284	0.9876	0.8059

Table B.12 Vector-wise accuracy of derivation, k=3, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.7701	0.9911	0.3333	0.9990	0.3922
4	0.8189	0.7834	1.0000	1.0000	0.7188	0.9332	0.4010	0.9689	0.4209
5	0.7202	0.6977	0.9437	0.9437	0.6900	0.8720	0.4342	0.9692	0.4217
6	0.6987	0.6712	0.9180	0.9180	0.6763	0.8488	0.4142	0.9641	0.4165
7	0.7196	0.6643	0.9230	0.9230	0.6911	0.8541	0.3837	0.9669	0.4355
8	0.7650	0.6825	0.9521	0.9521	0.7420	0.8506	0.3977	0.9576	0.4560
9	0.7711	0.6762	0.8984	0.8984	0.7339	0.8377	0.3942	0.9526	0.4501
10	0.8223	0.7057	0.9422	0.9422	0.7787	0.8561	0.3911	0.9634	0.4676

B.1.5 N=500, slip=guess=0.1

Table B.13 F-score of derivation, k=3, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8885	0.9960	0.3333	0.9994	0.7428
4	0.8951	0.8708	1.0000	1.0000	0.8665	0.9713	0.6060	0.9857	0.7666
5	0.8547	0.8294	0.9888	0.9887	0.8451	0.9471	0.6620	0.9787	0.7766
6	0.8422	0.8181	0.9691	0.9691	0.8213	0.9337	0.6831	0.9814	0.7877
7	0.8561	0.8250	0.9816	0.9816	0.8370	0.9369	0.6835	0.9718	0.8058
8	0.8813	0.8371	0.9893	0.9892	0.8608	0.9458	0.6986	0.9764	0.8089
9	0.9056	0.8479	0.9845	0.9844	0.8869	0.9456	0.7013	0.9769	0.8187
10	0.9180	0.8633	0.9803	0.9802	0.8951	0.9468	0.7092	0.9719	0.8224

Table B.14 Cell-wise accuracy of derivation, k=3, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.9134	0.9968	0.5556	0.9997	0.7570
4	0.9135	0.8949	1.0000	1.0000	0.8855	0.9759	0.6819	0.9876	0.7710
5	0.8687	0.8472	0.9895	0.9894	0.8581	0.9510	0.7008	0.9803	0.7681
6	0.8528	0.8335	0.9688	0.9688	0.8361	0.9371	0.7152	0.9823	0.7759
7	0.8575	0.8285	0.9799	0.9799	0.8400	0.9358	0.7038	0.9717	0.7881
8	0.8804	0.8377	0.9879	0.9879	0.8604	0.9444	0.7155	0.9771	0.7924
9	0.9014	0.8442	0.9826	0.9825	0.8829	0.9429	0.7159	0.9771	0.7993
10	0.9144	0.8599	0.9784	0.9783	0.8917	0.9441	0.7270	0.9718	0.8009

Table B.15 Vector-wise accuracy of derivation, k=3, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.7724	0.9898	0.3333	0.9990	0.4018
4	0.8221	0.7836	1.0000	1.0000	0.7219	0.9264	0.4221	0.9662	0.4244
5	0.7131	0.6777	0.9686	0.9683	0.6716	0.8572	0.3952	0.9472	0.4053
6	0.6753	0.6431	0.9062	0.9062	0.6444	0.8158	0.4033	0.9517	0.4130
7	0.6891	0.6287	0.9397	0.9397	0.6552	0.8167	0.3721	0.9210	0.4362
8	0.7382	0.6491	0.9647	0.9646	0.6957	0.8373	0.3838	0.9374	0.4424
9	0.7802	0.6566	0.9479	0.9476	0.7428	0.8328	0.3871	0.9374	0.4543
10	0.8103	0.6891	0.9383	0.9382	0.7550	0.8387	0.3983	0.9221	0.4563

B.1.6 N=500, slip=guess=0.2

Table B.16 F-score of derivation, k=3, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.8881	0.9946	0.3333	0.9992	0.7398
4	0.8521	0.8493	0.9667	0.9667	0.8310	0.9445	0.5033	0.9606	0.7709
5	0.7940	0.7847	0.9430	0.9025	0.7916	0.8818	0.5989	0.9500	0.7646
6	0.7972	0.7705	0.9453	0.8988	0.7792	0.8905	0.6341	0.9303	0.7906
7	0.8171	0.7839	0.9437	0.8894	0.7905	0.8828	0.6498	0.9448	0.7928
8	0.8393	0.7930	0.9544	0.8994	0.8140	0.8881	0.6620	0.9377	0.8015
9	0.8618	0.8071	0.9551	0.9037	0.8248	0.8873	0.6675	0.9413	0.8107
10	0.8747	0.8075	0.9529	0.9121	0.8288	0.8864	0.6802	0.9423	0.8138

Table B.17 Cell-wise accuracy of derivation, k=3, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.9126	0.9958	0.5556	0.9996	0.7538
4	0.8783	0.8757	0.9741	0.9741	0.8562	0.9535	0.6213	0.9683	0.7736
5	0.8192	0.8120	0.9484	0.9127	0.8147	0.8958	0.6713	0.9572	0.7577
6	0.8112	0.7886	0.9471	0.9052	0.7963	0.8950	0.6776	0.9357	0.7740
7	0.8211	0.7916	0.9427	0.8918	0.7993	0.8826	0.6809	0.9491	0.7757
8	0.8410	0.7980	0.9529	0.9012	0.8181	0.8864	0.6879	0.9413	0.7809
9	0.8586	0.8062	0.9532	0.9030	0.8233	0.8826	0.6879	0.9442	0.7912
10	0.8713	0.8059	0.9499	0.9097	0.8276	0.8814	0.7008	0.9446	0.7917

Table B.18 Vector-wise accuracy of derivation, k=3, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
3	1.0000	1.0000	1.0000	1.0000	0.7664	0.9870	0.3333	0.9987	0.3997
4	0.7559	0.7516	0.9613	0.9613	0.6779	0.8785	0.3574	0.9293	0.4299
5	0.6305	0.6172	0.8608	0.7816	0.6034	0.7530	0.3959	0.9039	0.3936
6	0.6049	0.5661	0.8712	0.7895	0.5736	0.7332	0.3706	0.8608	0.4130
7	0.6164	0.5607	0.8592	0.7633	0.5779	0.7083	0.3549	0.8825	0.4136
8	0.6637	0.5763	0.8913	0.7981	0.6103	0.7146	0.3521	0.8682	0.4232
9	0.6978	0.5860	0.8983	0.8110	0.6165	0.7035	0.3546	0.8753	0.4386
10	0.7212	0.5820	0.8773	0.8120	0.6241	0.6987	0.3588	0.8723	0.4398

B.2 Results of Q-matrix Derivation: 4-skill case

B.2.1 N=200, slip=guess=0.05

Table B.19 F-score of derivation, k=4, N=200, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8106	0.8704	0.2500	0.9996	0.6502
5	0.8436	0.8075	0.9969	0.9973	0.8066	0.8601	0.5124	0.9472	0.6808
6	0.8007	0.7667	0.9814	0.9813	0.7945	0.8571	0.5928	0.9367	0.6991
7	0.7913	0.7541	0.9676	0.9668	0.7587	0.8601	0.6361	0.9321	0.7170
8	0.8042	0.7628	0.9668	0.9655	0.7636	0.8624	0.6657	0.9245	0.7306
9	0.8012	0.7594	0.9553	0.9538	0.7626	0.8653	0.6667	0.9177	0.7461
10	0.8188	0.7650	0.9437	0.9402	0.7513	0.8741	0.6752	0.9066	0.7500
11	0.8236	0.7670	0.9545	0.9542	0.7699	0.8690	0.6868	0.9091	0.7635
12	0.8445	0.7845	0.9654	0.9686	0.7930	0.8743	0.6741	0.8979	0.7646
13	0.8161	0.7773	0.9381	0.9413	0.7878	0.8748	0.6983	0.9024	0.7685
14	0.8449	0.7945	0.9608	0.9642	0.7994	0.8811	0.6649	0.8947	0.7761
15	0.8376	0.7824	0.9424	0.9454	0.7883	0.8909	0.6519	0.8992	0.7767
16	0.8492	0.8016	0.9438	0.9490	0.7937	0.8897	0.6899	0.8902	0.7816
17	0.8637	0.8148	0.9697	0.9735	0.8149	0.8944	0.6747	0.8887	0.7881
18	0.8540	0.7979	0.9440	0.9455	0.8093	0.8883	0.6604	0.8893	0.7863
19	0.8409	0.7841	0.9184	0.9223	0.7867	0.8860	0.6880	0.8933	0.7886
20	0.8676	0.8081	0.9646	0.9594	0.8095	0.8849	0.6834	0.8919	0.7906

Table B.20 Cell-wise accuracy of derivation, k=4, N=200, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8850	0.9147	0.6250	0.9998	0.7199
5	0.9013	0.8796	0.9980	0.9982	0.8677	0.8946	0.6863	0.9668	0.7241
6	0.8599	0.8357	0.9877	0.9876	0.8509	0.8851	0.7163	0.9576	0.7295
7	0.8427	0.8150	0.9759	0.9753	0.8185	0.8800	0.7321	0.9524	0.7355
8	0.8449	0.8138	0.9737	0.9727	0.8148	0.8780	0.7459	0.9444	0.7404
9	0.8354	0.8030	0.9622	0.9610	0.8066	0.8750	0.7407	0.9383	0.7524
10	0.8466	0.8003	0.9528	0.9495	0.7922	0.8811	0.7447	0.9278	0.7524
11	0.8513	0.8050	0.9624	0.9621	0.8101	0.8752	0.7547	0.9277	0.7616
12	0.8604	0.8099	0.9673	0.9698	0.8187	0.8757	0.7365	0.9174	0.7598
13	0.8377	0.8071	0.9421	0.9445	0.8160	0.8783	0.7593	0.9201	0.7633
14	0.8584	0.8147	0.9641	0.9672	0.8203	0.8787	0.7285	0.9128	0.7677
15	0.8381	0.7867	0.9413	0.9443	0.7948	0.8834	0.7004	0.9160	0.7681
16	0.8592	0.8187	0.9466	0.9512	0.8128	0.8854	0.7487	0.9073	0.7709
17	0.8692	0.8257	0.9713	0.9754	0.8255	0.8897	0.7316	0.9049	0.7775
18	0.8569	0.8053	0.9440	0.9456	0.8194	0.8795	0.7168	0.9049	0.7749
19	0.8462	0.7961	0.9186	0.9228	0.8032	0.8801	0.7437	0.9089	0.7783
20	0.8769	0.8204	0.9669	0.9619	0.8263	0.8796	0.7410	0.9075	0.7793

Table B.21 Vector-wise accuracy of derivation, k=4, N=200, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.6418	0.6660	0.2500	0.9992	0.2465
5	0.7772	0.7278	0.9947	0.9952	0.6177	0.5958	0.3203	0.9350	0.2514
6	0.6674	0.6109	0.9563	0.9559	0.6076	0.5661	0.3230	0.8935	0.2460
7	0.6272	0.5574	0.9123	0.9106	0.5622	0.5551	0.3138	0.8777	0.2451
8	0.6253	0.5555	0.9063	0.9049	0.5576	0.5557	0.3257	0.8560	0.2526
9	0.5984	0.5257	0.8683	0.8660	0.5353	0.5392	0.3210	0.8385	0.2751
10	0.6144	0.5075	0.8413	0.8344	0.5081	0.5537	0.3356	0.8279	0.2672
11	0.6114	0.5205	0.8670	0.8676	0.5369	0.5347	0.3153	0.8268	0.2788
12	0.6589	0.5464	0.9031	0.9135	0.5656	0.5604	0.3125	0.8038	0.2783
13	0.6077	0.5260	0.8313	0.8375	0.5635	0.5471	0.3351	0.8075	0.2760
14	0.6540	0.5540	0.8862	0.8938	0.5701	0.5656	0.2853	0.7939	0.2938
15	0.6030	0.4985	0.8215	0.8326	0.4970	0.6022	0.2570	0.7990	0.2921
16	0.6500	0.5543	0.8375	0.8500	0.5582	0.5855	0.3109	0.7895	0.2977
17	0.6772	0.5618	0.9132	0.9250	0.5676	0.6018	0.2982	0.7750	0.3063
18	0.6507	0.5233	0.8347	0.8378	0.5698	0.5691	0.2708	0.7737	0.3035
19	0.6211	0.4848	0.7532	0.7649	0.5333	0.5696	0.3076	0.7873	0.3100
20	0.6861	0.5533	0.8967	0.8861	0.5711	0.5589	0.2994	0.7773	0.3105

B.2.2 N=200, slip=guess=0.1

Table B.22 F-score of derivation, k=4, N=200, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8153	0.8752	0.2500	0.9999	0.6545
5	0.8207	0.7865	0.9732	0.9732	0.7821	0.8400	0.4905	0.9317	0.6750
6	0.7659	0.7419	0.9428	0.9340	0.7428	0.8389	0.5813	0.9146	0.7019
7	0.7624	0.7339	0.9358	0.9244	0.7200	0.8418	0.6241	0.9049	0.7157
8	0.7591	0.7256	0.9209	0.9032	0.7087	0.8473	0.6422	0.8808	0.7314
9	0.7580	0.7226	0.8970	0.8836	0.7074	0.8492	0.6546	0.8740	0.7407
10	0.7619	0.7295	0.8900	0.8726	0.7128	0.8519	0.6645	0.8785	0.7427
11	0.7727	0.7281	0.8871	0.8697	0.7278	0.8569	0.6752	0.8690	0.7569
12	0.7887	0.7452	0.8799	0.8701	0.7393	0.8572	0.6790	0.8694	0.7605
13	0.7884	0.7463	0.8821	0.8710	0.7364	0.8651	0.6707	0.8755	0.7590
14	0.8048	0.7574	0.8787	0.8728	0.7484	0.8695	0.6746	0.8675	0.7711
15	0.8071	0.7588	0.8810	0.8723	0.7524	0.8703	0.6757	0.8560	0.7747
16	0.8122	0.7617	0.8717	0.8623	0.7580	0.8720	0.6843	0.8561	0.7807
17	0.8152	0.7663	0.8653	0.8606	0.7572	0.8748	0.6828	0.8607	0.7818
18	0.8232	0.7642	0.8636	0.8589	0.7656	0.8755	0.6728	0.8574	0.7807
19	0.8177	0.7643	0.8541	0.8541	0.7620	0.8795	0.6814	0.8509	0.7918
20	0.8241	0.7744	0.8608	0.8524	0.7672	0.8829	0.6740	0.8477	0.7939

Table B.23 Cell-wise accuracy of derivation, k=4, N=200, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8891	0.9182	0.6250	1.0000	0.7246
5	0.8880	0.8670	0.9830	0.9830	0.8551	0.8811	0.6880	0.9590	0.7221
6	0.8378	0.8221	0.9605	0.9546	0.8210	0.8733	0.7174	0.9441	0.7304
7	0.8208	0.8009	0.9518	0.9436	0.7908	0.8653	0.7277	0.9354	0.7343
8	0.8078	0.7845	0.9365	0.9226	0.7722	0.8635	0.7268	0.9139	0.7390
9	0.8020	0.7760	0.9133	0.9029	0.7666	0.8622	0.7329	0.9072	0.7444
10	0.7991	0.7759	0.9045	0.8899	0.7643	0.8614	0.7349	0.9089	0.7439
11	0.8042	0.7689	0.8995	0.8849	0.7718	0.8624	0.7400	0.9007	0.7542
12	0.8154	0.7812	0.8925	0.8840	0.7783	0.8622	0.7435	0.8988	0.7564
13	0.8082	0.7738	0.8905	0.8810	0.7682	0.8650	0.7300	0.9030	0.7539
14	0.8207	0.7811	0.8850	0.8807	0.7753	0.8674	0.7326	0.8939	0.7639
15	0.8193	0.7793	0.8855	0.8777	0.7747	0.8669	0.7311	0.8842	0.7650
16	0.8251	0.7813	0.8776	0.8694	0.7802	0.8688	0.7398	0.8819	0.7694
17	0.8264	0.7846	0.8715	0.8675	0.7784	0.8709	0.7390	0.8860	0.7706
18	0.8323	0.7797	0.8688	0.8651	0.7831	0.8697	0.7278	0.8811	0.7675
19	0.8272	0.7800	0.8606	0.8612	0.7795	0.8741	0.7366	0.8748	0.7773
20	0.8280	0.7845	0.8634	0.8557	0.7795	0.8746	0.7256	0.8716	0.7791

Table B.24 Vector-wise accuracy of derivation, k=4, N=200, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.6486	0.6789	0.2500	0.9998	0.2541
5	0.7489	0.6997	0.9519	0.9514	0.6070	0.5633	0.3239	0.9178	0.2487
6	0.6134	0.5789	0.8757	0.8646	0.5561	0.5465	0.3247	0.8772	0.2527
7	0.5796	0.5292	0.8624	0.8459	0.5016	0.5294	0.3134	0.8481	0.2520
8	0.5424	0.4943	0.8159	0.7909	0.4641	0.5315	0.2984	0.8154	0.2538
9	0.5229	0.4660	0.7428	0.7266	0.4510	0.5180	0.3087	0.7933	0.2613
10	0.5163	0.4677	0.7259	0.6984	0.4427	0.5173	0.3062	0.8013	0.2544
11	0.5293	0.4516	0.7213	0.6942	0.4612	0.5217	0.3152	0.7857	0.2705
12	0.5521	0.4764	0.7068	0.6949	0.4710	0.5247	0.3192	0.7755	0.2743
13	0.5386	0.4657	0.7083	0.6927	0.4536	0.5299	0.2952	0.7855	0.2598
14	0.5720	0.4828	0.7053	0.7044	0.4659	0.5434	0.3064	0.7663	0.2839
15	0.5709	0.4771	0.7063	0.6943	0.4659	0.5435	0.3038	0.7547	0.2823
16	0.5795	0.4778	0.6880	0.6778	0.4761	0.5486	0.3113	0.7450	0.2889
17	0.5822	0.4815	0.6767	0.6740	0.4704	0.5526	0.3059	0.7513	0.2971
18	0.5935	0.4724	0.6748	0.6716	0.4787	0.5468	0.2888	0.7375	0.2873
19	0.5863	0.4755	0.6634	0.6659	0.4687	0.5620	0.3020	0.7336	0.3120
20	0.5945	0.4861	0.6715	0.6547	0.4694	0.5691	0.2812	0.7189	0.3122

B.2.3 N=200, slip=guess=0.2

Table B.25 F-score of derivation, k=4, N=200, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8117	0.8639	0.2500	0.9993	0.6567
5	0.7837	0.7718	0.8960	0.8928	0.7443	0.7990	0.4117	0.9054	0.6660
6	0.7145	0.6993	0.8144	0.7898	0.6920	0.7880	0.4877	0.8588	0.6859
7	0.6873	0.6766	0.7638	0.7252	0.6612	0.7882	0.5316	0.8301	0.6980
8	0.6758	0.6631	0.7310	0.6900	0.6502	0.7815	0.5648	0.8154	0.7065
9	0.6683	0.6555	0.7079	0.6762	0.6452	0.7826	0.5790	0.7924	0.7172
10	0.6603	0.6507	0.6974	0.6673	0.6473	0.7830	0.6090	0.7911	0.7283
11	0.6600	0.6428	0.6812	0.6640	0.6455	0.7848	0.6086	0.7780	0.7363
12	0.6607	0.6469	0.6783	0.6632	0.6422	0.7867	0.6291	0.7671	0.7410
13	0.6572	0.6457	0.6666	0.6580	0.6402	0.7970	0.6274	0.7634	0.7459
14	0.6549	0.6433	0.6655	0.6560	0.6398	0.7907	0.6343	0.7623	0.7515
15	0.6597	0.6451	0.6639	0.6591	0.6424	0.7965	0.6406	0.7587	0.7552
16	0.6609	0.6475	0.6614	0.6563	0.6416	0.7980	0.6459	0.7739	0.7558
17	0.6601	0.6477	0.6597	0.6564	0.6444	0.7969	0.6486	0.7553	0.7635
18	0.6649	0.6492	0.6549	0.6511	0.6442	0.8038	0.6461	0.7538	0.7652
19	0.6627	0.6499	0.6496	0.6482	0.6455	0.8074	0.6543	0.7543	0.7663
20	0.6686	0.6541	0.6540	0.6492	0.6472	0.8105	0.6508	0.7464	0.7732

Table B.26 Cell-wise accuracy of derivation, k=4, N=200, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8862	0.9097	0.6250	0.9997	0.7276
5	0.8661	0.8587	0.9352	0.9332	0.8335	0.8525	0.6630	0.9457	0.7150
6	0.8024	0.7924	0.8705	0.8547	0.7871	0.8349	0.6740	0.9138	0.7194
7	0.7679	0.7616	0.8222	0.7965	0.7518	0.8220	0.6813	0.8918	0.7178
8	0.7521	0.7435	0.7898	0.7636	0.7350	0.8099	0.6876	0.8788	0.7194
9	0.7356	0.7268	0.7624	0.7435	0.7205	0.8020	0.6852	0.8587	0.7221
10	0.7272	0.7215	0.7539	0.7356	0.7203	0.8009	0.6994	0.8554	0.7306
11	0.7199	0.7090	0.7352	0.7267	0.7113	0.7977	0.6911	0.8421	0.7327
12	0.7182	0.7084	0.7304	0.7214	0.7056	0.7963	0.7016	0.8317	0.7352
13	0.7063	0.6988	0.7138	0.7088	0.6954	0.8005	0.6932	0.8277	0.7389
14	0.7020	0.6939	0.7107	0.7047	0.6920	0.7920	0.6968	0.8253	0.7429
15	0.7048	0.6943	0.7076	0.7056	0.6925	0.7966	0.7004	0.8199	0.7432
16	0.7038	0.6943	0.7040	0.7016	0.6900	0.7971	0.7050	0.8316	0.7452
17	0.6992	0.6901	0.6998	0.6980	0.6881	0.7937	0.7049	0.8150	0.7503
18	0.7001	0.6884	0.6921	0.6901	0.6851	0.7976	0.7008	0.8114	0.7502
19	0.6984	0.6891	0.6892	0.6887	0.6863	0.8025	0.7098	0.8128	0.7509
20	0.7075	0.6977	0.6968	0.6945	0.6897	0.8071	0.7086	0.8039	0.7562

Table B.27 Vector-wise accuracy of derivation, k=4, N=200, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.6457	0.6420	0.2500	0.9988	0.2551
5	0.6947	0.6711	0.8481	0.8456	0.5752	0.5083	0.2747	0.8872	0.2341
6	0.5398	0.5160	0.6893	0.6578	0.4984	0.4940	0.2818	0.8178	0.2367
7	0.4569	0.4366	0.5819	0.5281	0.4223	0.4723	0.2763	0.7710	0.2265
8	0.4232	0.3943	0.5008	0.4469	0.3833	0.4433	0.2622	0.7385	0.2230
9	0.3885	0.3705	0.4480	0.4148	0.3597	0.4273	0.2491	0.7065	0.2254
10	0.3670	0.3523	0.4266	0.3944	0.3531	0.4199	0.2494	0.6941	0.2341
11	0.3592	0.3357	0.3885	0.3793	0.3345	0.4174	0.2330	0.6779	0.2375
12	0.3544	0.3320	0.3781	0.3632	0.3229	0.4068	0.2516	0.6477	0.2467
13	0.3317	0.3151	0.3577	0.3466	0.3078	0.4203	0.2318	0.6550	0.2433
14	0.3232	0.3040	0.3437	0.3333	0.3061	0.3961	0.2420	0.6447	0.2491
15	0.3353	0.3094	0.3383	0.3376	0.3027	0.4084	0.2416	0.6338	0.2481
16	0.3355	0.3134	0.3309	0.3325	0.3033	0.4124	0.2557	0.6453	0.2498
17	0.3241	0.3027	0.3294	0.3244	0.2977	0.4051	0.2527	0.6277	0.2582
18	0.3280	0.3020	0.3128	0.3100	0.2885	0.4113	0.2484	0.6207	0.2618
19	0.3227	0.2983	0.3033	0.3060	0.2881	0.4263	0.2636	0.6199	0.2599
20	0.3395	0.3150	0.3023	0.3145	0.3000	0.4277	0.2714	0.6096	0.2712

B.2.4 N=500, slip=guess=0.05

Table B.28 F-score of derivation, k=4, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8297	0.8841	0.2500	0.9997	0.6523
5	0.8288	0.8095	1.0000	0.9999	0.8218	0.8561	0.5385	0.9632	0.6801
6	0.7673	0.7583	0.9804	0.9804	0.7884	0.8635	0.6004	0.9390	0.7073
7	0.7704	0.7596	0.9752	0.9751	0.7530	0.8628	0.6301	0.9312	0.7216
8	0.7850	0.7594	0.9650	0.9650	0.7588	0.8674	0.6499	0.9191	0.7298
9	0.7908	0.7612	0.9586	0.9586	0.7647	0.8686	0.6587	0.9136	0.7461
10	0.8061	0.7726	0.9676	0.9676	0.7885	0.8759	0.6628	0.9196	0.7559
11	0.8138	0.7697	0.9506	0.9506	0.7944	0.8725	0.6549	0.9071	0.7611
12	0.8221	0.7782	0.9438	0.9438	0.8045	0.8796	0.6736	0.9137	0.7639
13	0.8343	0.7958	0.9555	0.9555	0.8123	0.8860	0.6651	0.9103	0.7716
14	0.8453	0.7939	0.9567	0.9567	0.8170	0.8836	0.6672	0.9067	0.7757
15	0.8495	0.7969	0.9498	0.9497	0.8219	0.8886	0.6732	0.9065	0.7812
16	0.8487	0.8003	0.9514	0.9516	0.8265	0.8921	0.6581	0.9081	0.7814
17	0.8597	0.8026	0.9448	0.9448	0.8318	0.8952	0.6695	0.9020	0.7832
18	0.8725	0.8136	0.9616	0.9615	0.8414	0.9014	0.6646	0.8953	0.7850
19	0.8715	0.8119	0.9557	0.9557	0.8384	0.8988	0.6690	0.9013	0.7922
20	0.8701	0.8100	0.9507	0.9507	0.8324	0.9014	0.6759	0.8983	0.7922

Table B.29 Cell-wise accuracy of derivation, k=4, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8979	0.9254	0.6250	0.9999	0.7228
5	0.8929	0.8810	1.0000	1.0000	0.8788	0.8927	0.7017	0.9769	0.7256
6	0.8362	0.8298	0.9867	0.9867	0.8458	0.8900	0.7231	0.9584	0.7345
7	0.8281	0.8206	0.9816	0.9815	0.8137	0.8834	0.7271	0.9507	0.7371
8	0.8306	0.8113	0.9724	0.9724	0.8096	0.8833	0.7354	0.9405	0.7404
9	0.8299	0.8077	0.9659	0.9659	0.8095	0.8811	0.7377	0.9330	0.7500
10	0.8351	0.8093	0.9719	0.9719	0.8210	0.8831	0.7348	0.9369	0.7583
11	0.8341	0.7990	0.9545	0.9545	0.8199	0.8756	0.7222	0.9262	0.7592
12	0.8433	0.8069	0.9483	0.9483	0.8292	0.8826	0.7405	0.9304	0.7607
13	0.8479	0.8155	0.9573	0.9573	0.8298	0.8851	0.7276	0.9263	0.7670
14	0.8570	0.8128	0.9583	0.9583	0.8338	0.8818	0.7302	0.9226	0.7685
15	0.8595	0.8127	0.9512	0.9511	0.8361	0.8855	0.7342	0.9216	0.7736
16	0.8557	0.8128	0.9522	0.9524	0.8376	0.8875	0.7191	0.9229	0.7747
17	0.8664	0.8144	0.9449	0.9449	0.8431	0.8908	0.7299	0.9167	0.7741
18	0.8767	0.8223	0.9617	0.9617	0.8492	0.8963	0.7249	0.9111	0.7737
19	0.8765	0.8215	0.9548	0.9548	0.8469	0.8940	0.7287	0.9145	0.7810
20	0.8747	0.8184	0.9516	0.9516	0.8405	0.8961	0.7352	0.9120	0.7814

Table B.30 Vector-wise accuracy of derivation, k=4, N=500, s=g=0.05

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.6660	0.7021	0.2500	0.9994	0.2404
5	0.7630	0.7331	1.0000	0.9998	0.6391	0.5837	0.3323	0.9428	0.2558
6	0.6221	0.6030	0.9466	0.9466	0.6010	0.5863	0.3367	0.8961	0.2562
7	0.5896	0.5667	0.9286	0.9283	0.5463	0.5651	0.3119	0.8724	0.2513
8	0.5894	0.5349	0.8983	0.8983	0.5268	0.5636	0.3196	0.8558	0.2540
9	0.5824	0.5390	0.8793	0.8793	0.5345	0.5599	0.3196	0.8349	0.2666
10	0.6012	0.5345	0.8992	0.8992	0.5623	0.5585	0.3070	0.8443	0.2744
11	0.6061	0.5384	0.8586	0.8586	0.5495	0.5495	0.3051	0.8268	0.2789
12	0.6089	0.5401	0.8199	0.8199	0.5836	0.5629	0.3070	0.8290	0.2773
13	0.6257	0.5499	0.8552	0.8552	0.5863	0.5800	0.2961	0.8239	0.2832
14	0.6442	0.5433	0.8610	0.8610	0.5919	0.5635	0.3000	0.8164	0.2916
15	0.6467	0.5432	0.8453	0.8451	0.5947	0.5744	0.3096	0.8098	0.2993
16	0.6428	0.5449	0.8532	0.8535	0.5949	0.5851	0.2802	0.8113	0.3010
17	0.6612	0.5417	0.8435	0.8435	0.5981	0.5928	0.2935	0.8085	0.3015
18	0.6936	0.5648	0.8872	0.8871	0.6199	0.6165	0.2819	0.7903	0.3025
19	0.6870	0.5602	0.8647	0.8647	0.6104	0.6066	0.2845	0.8009	0.3150
20	0.6857	0.5514	0.8600	0.8600	0.5992	0.6167	0.2934	0.7955	0.3141

B.2.5 N=500, slip=guess=0.1

Table B.31 F-score of derivation, k=4, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8272	0.8813	0.2500	0.9997	0.6513
5	0.8005	0.7905	0.9966	0.9966	0.8085	0.8468	0.5041	0.9380	0.6773
6	0.7598	0.7512	0.9836	0.9816	0.7540	0.8418	0.5846	0.9106	0.7053
7	0.7473	0.7376	0.9741	0.9703	0.7209	0.8496	0.6136	0.8928	0.7135
8	0.7527	0.7300	0.9648	0.9594	0.7233	0.8530	0.6324	0.8944	0.7338
9	0.7667	0.7360	0.9615	0.9564	0.7359	0.8557	0.6487	0.8898	0.7425
10	0.7755	0.7388	0.9395	0.9338	0.7455	0.8526	0.6537	0.8718	0.7567
11	0.7845	0.7491	0.9456	0.9410	0.7627	0.8569	0.6521	0.8842	0.7557
12	0.8047	0.7610	0.9520	0.9490	0.7817	0.8722	0.6623	0.8957	0.7584
13	0.8162	0.7693	0.9469	0.9435	0.7920	0.8702	0.6671	0.8735	0.7665
14	0.8357	0.7790	0.9616	0.9601	0.8027	0.8773	0.6665	0.8702	0.7746
15	0.8271	0.7776	0.9499	0.9488	0.8031	0.8838	0.6607	0.8784	0.7765
16	0.8422	0.7815	0.9517	0.9502	0.8063	0.8821	0.6715	0.8646	0.7825
17	0.8465	0.7897	0.9482	0.9476	0.8133	0.8844	0.6682	0.8566	0.7848
18	0.8648	0.7961	0.9609	0.9597	0.8249	0.8914	0.6593	0.8604	0.7892
19	0.8634	0.7952	0.9425	0.9420	0.8208	0.8920	0.6678	0.8629	0.7962
20	0.8648	0.7979	0.9485	0.9488	0.8184	0.8966	0.6654	0.8553	0.7957

Table B.32 Cell-wise accuracy of derivation, k=4, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8971	0.9221	0.6250	0.9999	0.7215
5	0.8747	0.8691	0.9978	0.9978	0.8725	0.8869	0.6939	0.9624	0.7216
6	0.8313	0.8264	0.9890	0.9878	0.8240	0.8736	0.7150	0.9413	0.7318
7	0.8094	0.8039	0.9804	0.9778	0.7904	0.8731	0.7171	0.9256	0.7301
8	0.8028	0.7873	0.9719	0.9677	0.7813	0.8690	0.7200	0.9239	0.7420
9	0.8091	0.7860	0.9674	0.9633	0.7858	0.8678	0.7278	0.9181	0.7463
10	0.8088	0.7817	0.9471	0.9421	0.7875	0.8614	0.7268	0.9021	0.7550
11	0.8135	0.7865	0.9517	0.9479	0.7982	0.8641	0.7233	0.9099	0.7539
12	0.8241	0.7888	0.9558	0.9531	0.8066	0.8748	0.7264	0.9196	0.7574
13	0.8349	0.7972	0.9500	0.9469	0.8165	0.8729	0.7316	0.8991	0.7585
14	0.8495	0.8013	0.9634	0.9619	0.8221	0.8778	0.7287	0.8948	0.7654
15	0.8380	0.7956	0.9503	0.9493	0.8190	0.8820	0.7221	0.9012	0.7674
16	0.8516	0.7989	0.9529	0.9516	0.8215	0.8802	0.7304	0.8890	0.7707
17	0.8532	0.8035	0.9477	0.9472	0.8260	0.8807	0.7269	0.8820	0.7721
18	0.8684	0.8056	0.9599	0.9588	0.8327	0.8854	0.7161	0.8831	0.7754
19	0.8671	0.8052	0.9425	0.9422	0.8293	0.8872	0.7243	0.8849	0.7829
20	0.8673	0.8059	0.9479	0.9481	0.8257	0.8910	0.7216	0.8790	0.7818

Table B.33 Vector-wise accuracy of derivation, k=4, N=500, s=g=0.1

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.6730	0.6926	0.2500	0.9994	0.2541
5	0.7156	0.7027	0.9952	0.9952	0.6366	0.5741	0.3333	0.9273	0.2489
6	0.6001	0.5895	0.9574	0.9540	0.5678	0.5493	0.3217	0.8755	0.2583
7	0.5430	0.5371	0.9244	0.9184	0.4953	0.5512	0.3047	0.8470	0.2435
8	0.5316	0.4938	0.8941	0.8837	0.4747	0.5370	0.2984	0.8291	0.2562
9	0.5470	0.4918	0.8835	0.8730	0.4901	0.5323	0.3012	0.8175	0.2579
10	0.5366	0.4755	0.8216	0.8092	0.4869	0.5129	0.3012	0.7942	0.2666
11	0.5459	0.4855	0.8350	0.8268	0.5037	0.5224	0.2977	0.7984	0.2645
12	0.5775	0.4980	0.8612	0.8556	0.5311	0.5606	0.3014	0.8180	0.2689
13	0.5932	0.5070	0.8372	0.8300	0.5423	0.5533	0.3069	0.7749	0.2771
14	0.6319	0.5204	0.8864	0.8839	0.5579	0.5696	0.2980	0.7647	0.2838
15	0.6043	0.5077	0.8520	0.8502	0.5503	0.5793	0.2845	0.7816	0.2869
16	0.6347	0.5184	0.8623	0.8600	0.5571	0.5762	0.2956	0.7554	0.2934
17	0.6412	0.5257	0.8464	0.8461	0.5699	0.5804	0.2891	0.7530	0.2951
18	0.6812	0.5323	0.8826	0.8810	0.5859	0.5927	0.2691	0.7541	0.3071
19	0.6703	0.5296	0.8343	0.8338	0.5729	0.5978	0.2796	0.7583	0.3194
20	0.6712	0.5295	0.8525	0.8534	0.5670	0.6113	0.2728	0.7390	0.3172

B.2.6 N=500, slip=guess=0.2

Table B.34 F-score of derivation, k=4, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8272	0.8774	0.2500	0.9999	0.6508
5	0.7823	0.7740	0.9453	0.9445	0.7599	0.8002	0.3975	0.9077	0.6713
6	0.7196	0.7214	0.8988	0.8532	0.7076	0.7857	0.4945	0.8498	0.6981
7	0.6964	0.6819	0.8702	0.7751	0.6766	0.7996	0.5274	0.8269	0.7068
8	0.6904	0.6709	0.8342	0.7304	0.6626	0.7887	0.5658	0.8031	0.7198
9	0.6967	0.6736	0.7989	0.7104	0.6660	0.7919	0.5848	0.8128	0.7274
10	0.7008	0.6746	0.7760	0.7037	0.6713	0.7867	0.6124	0.7868	0.7372
11	0.7058	0.6746	0.7696	0.7006	0.6782	0.7977	0.6011	0.7986	0.7432
12	0.7082	0.6764	0.7599	0.7053	0.6815	0.8006	0.6111	0.7816	0.7577
13	0.7154	0.6833	0.7536	0.7134	0.6864	0.8044	0.6143	0.7788	0.7633
14	0.7166	0.6870	0.7482	0.7143	0.6906	0.8080	0.6196	0.7808	0.7609
15	0.7208	0.6853	0.7372	0.7133	0.6943	0.8094	0.6295	0.7751	0.7692
16	0.7269	0.6926	0.7438	0.7222	0.7015	0.8135	0.6319	0.7764	0.7685
17	0.7358	0.7018	0.7396	0.7261	0.7045	0.8165	0.6374	0.7736	0.7748
18	0.7324	0.6968	0.7321	0.7207	0.7063	0.8157	0.6382	0.7742	0.7766
19	0.7416	0.7042	0.7317	0.7214	0.7134	0.8234	0.6412	0.7753	0.7819
20	0.7470	0.7064	0.7254	0.7203	0.7152	0.8270	0.6426	0.7715	0.7841

Table B.35 Cell-wise accuracy of derivation, k=4, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.8968	0.9198	0.6250	1.0000	0.7195
5	0.8632	0.8590	0.9653	0.9648	0.8419	0.8530	0.6532	0.9484	0.7188
6	0.8071	0.8083	0.9297	0.8988	0.7979	0.8350	0.6807	0.9091	0.7264
7	0.7723	0.7624	0.9002	0.8313	0.7584	0.8306	0.6759	0.8902	0.7247
8	0.7564	0.7448	0.8671	0.7898	0.7383	0.8148	0.6854	0.8687	0.7295
9	0.7583	0.7438	0.8349	0.7713	0.7382	0.8153	0.6897	0.8730	0.7334
10	0.7592	0.7421	0.8149	0.7653	0.7396	0.8082	0.7072	0.8520	0.7375
11	0.7503	0.7284	0.7994	0.7494	0.7313	0.8071	0.6841	0.8574	0.7401
12	0.7494	0.7267	0.7901	0.7508	0.7306	0.8082	0.6885	0.8416	0.7514
13	0.7510	0.7279	0.7823	0.7532	0.7295	0.8080	0.6876	0.8374	0.7538
14	0.7470	0.7264	0.7732	0.7492	0.7284	0.8087	0.6872	0.8366	0.7509
15	0.7493	0.7225	0.7626	0.7461	0.7289	0.8083	0.6939	0.8319	0.7577
16	0.7530	0.7272	0.7674	0.7514	0.7339	0.8114	0.6954	0.8312	0.7561
17	0.7602	0.7351	0.7642	0.7548	0.7353	0.8145	0.6996	0.8280	0.7619
18	0.7564	0.7298	0.7570	0.7489	0.7360	0.8125	0.7006	0.8271	0.7636
19	0.7601	0.7309	0.7532	0.7452	0.7378	0.8174	0.6993	0.8276	0.7681
20	0.7642	0.7325	0.7474	0.7447	0.7387	0.8208	0.7017	0.8232	0.7685

Table B.36 Vector-wise accuracy of derivation, k=4, N=500, s=g=0.2

J	k-means(ℓ_2)	k-means(ℓ_1)	k-medoids(ℓ_2)	k-medoids(ℓ_1)	MMB	HC	LASSO	NMF	ALS
4	1.0000	1.0000	1.0000	1.0000	0.6742	0.6836	0.2500	0.9998	0.2459
5	0.6898	0.6855	0.9120	0.9111	0.5930	0.5112	0.2675	0.8953	0.2430
6	0.5396	0.5432	0.8061	0.7499	0.5203	0.5022	0.2888	0.8290	0.2469
7	0.4694	0.4474	0.7357	0.6058	0.4334	0.4962	0.2792	0.8006	0.2326
8	0.4240	0.3979	0.6580	0.5103	0.3888	0.4597	0.2724	0.7633	0.2370
9	0.4313	0.3987	0.5843	0.4729	0.3869	0.4567	0.2574	0.7589	0.2377
10	0.4340	0.3966	0.5452	0.4612	0.3980	0.4271	0.2759	0.7433	0.2398
11	0.4263	0.3726	0.5268	0.4322	0.3724	0.4304	0.2376	0.7410	0.2463
12	0.4212	0.3701	0.5137	0.4334	0.3764	0.4365	0.2433	0.7218	0.2669
13	0.4272	0.3740	0.4966	0.4436	0.3737	0.4274	0.2421	0.7118	0.2648
14	0.4148	0.3706	0.4816	0.4373	0.3670	0.4352	0.2343	0.7109	0.2569
15	0.4247	0.3652	0.4598	0.4257	0.3743	0.4320	0.2536	0.7002	0.2670
16	0.4310	0.3763	0.4695	0.4346	0.3804	0.4415	0.2458	0.7013	0.2645
17	0.4479	0.3941	0.4609	0.4438	0.3889	0.4373	0.2590	0.6994	0.2758
18	0.4429	0.3814	0.4444	0.4333	0.3895	0.4366	0.2610	0.6907	0.2792
19	0.4504	0.3840	0.4441	0.4271	0.3880	0.4480	0.2572	0.6912	0.2928
20	0.4575	0.3846	0.4309	0.4239	0.3935	0.4559	0.2531	0.6934	0.2880