Exploring the feasibility of using artificial neural networks for cognitive diagnostic modeling: two case studies

by

Kang Xue

(Under the Direction of Laine P. Bradshaw)

Abstract

As a new research area, Artificial Neural Networks (ANNs) begins to attract some research attention for cognitive diagnostic classification. In my dissertation, we explore the feasibility of using ANNs for CDM. The dissertation consists of two research studies. In the first study, we firstly designed an unsupervised learning ANN for attribute estimation which does not rely on a specific assumption of item response function and just requires partial Q-matrix information (simple items' q-vectors); secondly, we proposed a Q-matrix reconstruction method using K-means clustering algorithms to correct or reconstruct the mis-specified or missing elements of the Q-matrix. In the second case study, we combined ANN with a semi-supervised learning method, the Co-Training method. To hold the two assumptions of successfully applying Co-Training, we used two theoretical diagnostic classification models, DINA and DINO models, as the two classifiers. In these two research studies, we systematically describe how to construct ANN for diagnostic classification and deal with the issues and challenges in the existed research studies. In the designed simulated study,we test the two method under different test conditions and make comparison with several widely used theoretical diagnostic classification models. The experimental results show the advantages of the proposed methods in dealing with the noises contained in assessments, such as low diagnostic quality, misspecified elements in Q-matrix. We also discuss the limits of the proposed methods and some future research directions of using ANN and machine learning for CDM and psychometrics.

Index words: Cognitive Diagnostic Modeling, Artificial Neural Networks, Unsupervised Leaning, Semi-Supervised Leaning, Q-Matrix Reconstruction

Exploring the feasibility of using artificial neural networks for cognitive diagnostic modeling: two case studies

by

Kang Xue

B.S., Beijing Institute of Technology, China, 2006
PH.D., Beijing Institute of Technology, China, 2013

A Dissertation Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the
Degree

Doctor of Philosophy

Athens, Georgia

2020

Exploring the feasibility of using artificial neural networks for cognitive diagnostic modeling: two case studies

by

Kang Xue

Major Professor:	Laine P. Bradshaw
Committee:	April Galyardt
	Seock-Ho Kim
	Shiyu Wang

Electronic Version Approved:

Ron Walcott
Interim Dean of the Graduate School
The University of Georgia
August 2020

Acknowledgments

I would like to thank to my PhD advisors, Dr. Laine Bradshaw, for supporting me during these past three years. Laine has been supportive and has given me the freedom to pursue research project I am interested and gave me lots of opportunities in her research projects to help me understand the theory and application in real psychometric problems. She has also provided insightful discussions about the research and the dissertation.

I am also very grateful to my former PhD advisor, Dr. April Galyardt, for her support and advisory in my first two years at the University of Georgia. April also helped me build up academic network with other famous researchers in the field educational data mining. She provided lots of constructive suggestions in my dissertation and my future research. I also have to thank the members of my PhD committee, Drs. Seock-Ho Kim and Shiyu Wang for their helpful career advice and suggestions in general.

I especially thank my parents and parents in law. They have sacrificed their retire lives to support me during these five years. I love them so much, and my wife and I would not have made it this far without them.

Special thanks to my wife Kuo and our sons, Edgar and Aiden. Although it is truly a serious challenge for two PhD students with two little monsters, we make it finally with our cooperation and mutual support in these five years. Also thanks to Edgar and Aiden for bringing us hope and happiness.

2020 is really a tough year because of the Covid-19. I deeply appreciate everyone who is helping people through this pandemic.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As a field of study concerned with the theory and technique of psychological measurement, psychometrics is concerned with the objective measurement of skills and knowledge, abilities, attitudes, personality traits, and educational achievement through designed assessment (Aiserman et al., 1964). Over the past several decades, researchers have developed a number of different statistical functions or theoretic psychometric models under different measurement theories. These theoretic psychometric models are widely used for analyzing students' responses in both assessment and virtual learning environment to determine both item parameters and person parameters (e.g., latent trait, latent class).

1.1 Theoretic psychometrics

Classical test theory (CTT; Crocker and Algina, 1986) is a traditional test theory which assumes a test taker's observed test score is the sum of the true test score and the error. The higher test score means a higher ability, and test score is a continuous variable. CTT treats the test takers with the same observed test score as having equal ability because CTT assumes each item has the same difficulty in a single assessment. Rooted in CTT, item response theory (IRT; Embretson and Reise, 2013), also known as the latent trait theory (Levine, 1982), refers to a family of mathematical models that attempt to explain the relationship between latent traits (unobservable characteristic or attribute) and their manifestations (i.e., observed outcomes, responses or performance) using different statistical functions (e.g., Rasch Model, 2-Parameter IRT model, 3-Parameter IRT model). In contrast to CTT for which results cannot be compared under two sample groups, the IRT models are not sample dependent and can locate test takers on the same continuous scale. According to this advantage, IRT models are widely

used for measurement for standardized assessment and data analysis in visual learning environments (VLEs; Britain and Liber, 2004), a Web-based platform which delivers learning materials via a digital space.

Cognitive Diagnostic Modeling (CDM; J. Templin, Henson, et al., 2010) is an area of psychometric research that has seen substantial growth over the past decade. It has been receiving more attention in many assessment situations because it has the ability to provide a finer evaluation of the examinees' trait instead of a simple overall score. In contrast to IRT, which provides a continuous unidimensional trait for each examinee, as a part of latent class theory, CDM uses diagnostic classification models (DCMs) to assign examinees into multiple latent groups by determining whether they have mastered a number of attributes. Recent studies have shown that DCMs had uniformly greater reliability for examinees' latent attribute estimation compared to IRT under the same test length condition (J. Templin & Bradshaw, 2013). In addition, the latent classification results are more interpretable than using a continuous latent variable for students, teachers and education administrators.

1.2 Machine learning and deep learning

The technological changes across learning, instruction and assessment start to bring machine learning techniques into psychometrics. For example, IRT psychometric models are usually based upon logistic regression techniques which are used to be popular in solving classification problem in machine learning (Martınez-Plumed et al., 2016).

Machine learning (Bishop, 2006) is an application of artificial intelligence (AI) that provides computers/systems the ability to learn and complete tasks without explicitly being programmed. Instead of inference, machine learning provides a series of algorithms and methods that focus on prediction. Generally, machine learning algorithms are categorized as supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning (Chapelle et al., 2009) according to different kinds of data and tasks. Generally, in the machine learning field, there are two kinds of data: labeled data and unlabeled data. Typically, unlabeled data consists of samples of raw data that you can obtain relatively easily from the world (e.g., a student's raw score, or a student's response pattern). Labeled data typically takes a set of unlabeled data and augments each piece of that unlabeled data with some sort of meaningful label, such as the latent class and the latent trait of a student. Supervised learning algorithms (e.g., Lasso logistic regression, random forests) can be learned from the labeled data and applied to new unlabeled data to predict future events by

learning through labeled examples in the past; in contrast, unsupervised learning algorithms (e.g., K-means, Principle Components Analysis) are to explore the unlabeled data and draw inferences from datasets to describe hidden structure (e.g., no labeled past data is available or provided); semi-supervised learning algorithms lay between supervised and unsupervised learning and use unlabeled data in conjunction with a small amount of labeled data to produce considerable improvement in model training accuracy and robustness.

As a subfield of machine learning, deep learning methods are widely applied to extract latent variables or features from the input distribution based on artificial neural networks (ANNs). An ANN is a computational system inspired by biological neural networks (LeCun et al., 2015). In the last several years, the artificial intelligence systems which rely on deep learning achieved impressive achievements in different research fields, such as computer vision, natural language processing and speech recognition. In educational research area, deep learning has been applied for different tasks, such as automatic item generation (AIG; von Davier, 2018), automated scoring (Taghipour & Ng, 2016), and item characteristics prediction (Xue et al., 2020).

Generally, due to the characteristics of deep learning, there are three potential advantages to applying deep learning techniques to psychometric applications. More details of deep learning techniques will be described in Chapter 2.

1. Deep learning techniques are used to exploit different latent variables concerned with different psychometric theories from the observed distribution, often at multiple levels;

2. Deep learning has proven to be a good solution for approximate computing (Hanin, 2019; Lu et al., 2017), which is a promising computation technique that relies on the ability of many systems to tolerate some loss of quality or optimality in computed results;

3. As a subfield of machine learning, deep learning based psychometric methods are also expected to be robust to the noises contained in the data when explaining the training dataset and doing prediction on new dataset.

In last several years, some types of ANNs, the basic computational systems in deep learning, were used for psychometrics, especially for cognitive diagnostic modeling. However, all the existing methods introduced ANNs as a computational approach in isolated application of psychometrics compared with the theoretic psychometric models.

1.3 Deep learning-based computational psychometric methods

Due to the development of techniques in instruction and assessment over past decade, large data sets and high dimensional data have become available for educational research. In contrast to traditional assessment data for theoretic psychometric models, the new data contains more noise and the prior knowledge of the data (e.g., for CDM, Q-matrix which represents the relationships between attributes and items) might be incomplete or mis-specified. In order to effectively apply theoretic psychometric models, statistical testing (e.g., ANOVA) was required to determine the item response function. Machine learning and deep learning could be another option to improve the robustness of the data exploration from such kinds of data and also extract new abstractive information which might be ignored by human.

In this research, to explore the feasibility of using deep learning techniques to extract latent person variables (e.g., latent trait, latent class) concerned by psychometrics, a family of Deep Learning-based Computational Psychometric Methods (DLCPMs) were proposed. To explain data for psychometrics and prediction on new dataset simultaneously, all DLCPMs in this research integrated deep learning and theoretic psychometrics in one system. DLCPMs could be applied under either unsupervised learning or semi-supervised learning frameworks by modifying the deep learning architecture. The unsupervised DLCPMs are used to do data exploration for further psychometric applications, and the semi-supervised DLCPMs are used to improve the performance and robustness of applying psychometrics. Generally, DLCPMs consist of three parts: *observed inputs*, *feature extraction* (or *latent variable extraction*), and *targeting*.

Observed inputs refers to the observed data such as students' responses to items in a designed assessment or interactions within an online learning environment, such as raw text, log data and sequential data. The *feature extraction* refers to the process of converting the observed response pattern to latent person variables that psychometric models are concerned with using deep learning techniques. For IRT, this latent variable refers to students' abilities (continuous variables) and for CDM the latent variable is attribute profiles (discrete variables). To extract different types of latent variables from the inputs, the structures of the deep learning architecture differ by the feature hierarchy: in deep learning, different features are learned at different levels from the same observation. We will discuss the feature hierarchy in detail in Chapter 2. The third part, *targeting*, has to do with both training and explaining. Like typical machine learning tasks, the process of training is to estimate appropriate pa-

rameters contained in the deep learning architecture for future prediction on a new dataset; the process of explaining is to make the latent variables exploited by the deep learning-based feature extraction have the same interpretation as the theoretic psychometric models.

In the experimental test, unsupervised deep learning algorithms and semi-supervised deep learning algorithms will be applied to extract the person parameters (i.e., attribute profiles, latent classes and latent trait) under different psychometric theories. The proposed deep learning-based computational psychometric methods (DLCPMs) could be used to extract both continuous latent variables (i.e., latent trait) and discrete latent variables (i.e., latent class, latent attribute mastery status). In Chapter 3 and 4, two applications using DLCPMs for Cognitive Diagnostic Modeling (CDM) are described in detail.

Chapter 2

Introduction

In this chapter, we present an introduction and overview of the theories and existing literature related to this research. This chapter will cover the following three parts: some basic concepts of cognitive diagnostic modeling (CDM) and item response theory (IRT); unsupervised learning and semi-supervised learning in machine learning; the feature hierarchy and approximate computing of deep learning. The chapter provides a theoretical foundation of the proposed DLCPMs.

2.1 Theoretic Psychometric Models

2.1.1 Cognitive Diagnostic Modeling

The purpose of cognitive diagnostic modeling (CDM) or diagnostic measurement is to provide students' knowledge mastery states through their responses to items from carefully designed assessments. Because of the ability to provide educators diagnostic feedback from students' assessment result, CDM have been the focus of much research in the last decade. Various types of diagnostic classification models (DCMs), such as the deterministic inputs, noisy "and" gate (DINA; Junker and Sijtsma, 2001), the reparametrized unified model/fusion model (RUM; Hartz, 2002) and the log-linear cognitive diagnosis model (LCDM; Henson et al., 2009), are designed based on different cognitive theories or hypotheses about how attributes behave, or interact, to produce individual item responses.

Attribute Profile and Latent Class

In CDM, an attribute profile is a A-dimensional vector, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_A]'$, for which the ath element indicates the ath attribute, for $a \in \{1, 2, ..., A\}$.

In most DCMs, the attribute mastery levels are dichotomous, where $\alpha_a = 1$ indicates mastery of Attribute a and $\alpha_a = 0$ indicates non-mastery of Attribute a. The number of latent classes C equals to 2^A, and students within a latent class c have same attribute profile α_c.

Q-matrix

Theoretically, the specification of which attributes are measured by each item is done numerically in a table called the Q-matrix (Tatsuoka, 1983). DCMs are specified based on a Q-matrix. A Q-matrix traditionally contains the items in the rows and the attributes in the columns. 1s and 0s in the Q-matrix indicates whether or not an attribute is measured by an item. Also taking the math cognitive diagnostic assessment as an example, a sample Q-matrix is shown in Table 2.1. This Q-matrix shows the relationship between four items and four attributes (add, subtract, multiply and divide). The first three items are simple structure items or simple items because each of them only measures single attribute. The last item ($16 - 2 \times 3 =?$) is a complex structure item or complex item because it measures more than one attribute. Each row of a Q-matrix is called a q-vector which indicates which attribute(s) the item is designed to measure.

Table 2.1: An example of Q-matrix.

Item	Add	Subtract	Multiply	Divide
$4 + 4 =?$	1	0	0	0
$12/2 =?$	0	0	0	1
$8 \times 2 =?$	0	0	1	0
16-2$\times$3 $=?$	0	1	1	0

A Q-matrix shows the relationship between four items and four attributes.

In addition, Q-matrix can also indicate the hierarchical structure of attributes intuitively. The attribute hierarchies represent the hypotheses of the attribute dependencies in the population of examinees. For example, if an attribute hierarchy specified that mastery of Attribute 1 is prerequisite to mastery of Attribute 2, in other word, Attribute 1 is generally acquired by population before Attribute 2. In the Q-matrix, if an item measures Attribute 2, it must measure Attribute 1 at the same time. Whereas the reverse is not true. It means that there are only 3 types of q-vectors indicating the relationship between items and two attributes (Attribute 1 and Attribute 2) in the Q-matrix: $(0, 0)$ refers to that the item measures none of the two attributes; $(1, 0)$ refers to that the item measures only Attribute 1; $(1, 1)$ means the item measures both two attributes. However, the q-vector pattern $(0, 1)$ cannot be observed in the Q-matrix.

Although Q-matrices are often designed carefully by assessment experts, some existing research and their experimental results have been shown that Q-matrices constructed by content experts do not always reflect the relationship precisely and may require empirically-driven modifications (Bradshaw et al., 2014; Tjoe & de la Torre, 2014). The misspecification can hardly be ignored, and several researchers have already shown the effects of the Q-matrix misspecification on different types of diagnostic classification models from experiments (Kunina-Habenicht et al., 2012; R. Liu et al., 2017; Madison & Bradshaw, 2015; A. A. Rupp & Templin, 2007). A number of studies have proposed automated methods to search or correct the Q-matrix with some constraints: Most of them require knowing the parametric item response function (Johnson, 2009), work only under conditions where an assessment contains a small number of items (Desmarais, 2012), or allow refinement to the Q-matrix in which only a small proportion of elements are mis-specified or missing (Chiu, 2013).

Typical Diagnostic Classification Models

There are two types of latent variable combinations in DCMs: (1) noncompensatory latent variable models where a low value on one latent variable cannot be compensated by a high value on another latent variable, and (2) compensatory latent variable models where a low value on one latent variable can be compensated for by a high value on another latent variable.

Deterministic inputs, noisy "and" gate model (DINA) A simple example of noncompensatory models is the DINA model (Junker & Sijtsma, 2001). The item response function of the DINA model for the ith item and latent class c is defined in terms of two parameters, slipping parameters s_i and guessing parameters g_i:

$$\pi_{i,c} = P(X_{i,c} = 1|\boldsymbol{\alpha}_c) = (1 - s_i)^{\eta_{i,c}} g_i^{1-\eta_{i,c}} \tag{2.1}$$

where $\boldsymbol{\alpha}_c$ is the attribute pattern for latent class c, and $\eta_{i,c} = \prod_{a=1}^{A} \alpha_{c,a}^{q_{i,a}}$ (and-gate) indicates whether examinees in latent class c have mastered all attributes measured by ith item ($\eta_{i,c} = 1$) or not ($\eta_{i,c} = 0$) based on conjunctive condensation rule. In current research studies, DINA model is widely chosen for simulation test because of the simplicity. For each item, there are only two potential $P(X_{i,c} = 1)$ for $\eta_{i,c} = 1$ and $\eta_{i,c} = 0$ respectively. The probability that an examinee can answer an item correctly drops severely if any of the measured attributes are not mastered. Due to the simplicity of DINA, a large

number of simulated studies for CDMs were conducted based on the DINA assumption (Chiu et al., 2009; Cui et al., 2012; Cui et al., 2016; De La Torre, 2008).

Deterministic, noisy "or" gate model (DINO) In contrast to noncompensatory models, DINO is a type of compensatory model (J. L. Templin & Henson, 2006). Like the DINA model, there are also two item parameters, slipping parameters s_i and guessing parameter g_i, contained in the mathematic representation of DINO model, but DINO model uses or-gate $\omega_{i,c}$ instead of and-gate $\eta_{i,c}$ in DINA:

$$\pi_{i,c} = P(X_{i,c} = 1|\boldsymbol{\alpha}_c) = (1 - s_i)^{\omega_{i,c}} g_i^{1-\omega_{i,c}} \tag{2.2}$$

where $\omega_{i,c} = 1 - \prod_{a=1}^{A}(1 - a_{c,a})^{q_{i,a}}$ indicates whether examinees in latent class c have mastered at least one of the attributes measured by ith item ($\omega_{i,c} = 1$) or not ($\omega_{i,c} = 0$).

Because different models, such as DINA and DINO, require different assumptions and have various levels of complexity, researchers have to select the appropriate model before conducting an analysis. Inappropriate model selection will influence the analysis results. According to this issue, some more general models are proposed (von Davier, 2005).

Log-linear cognitive diagnosis model (LCDM) As a general DCM, the log-linear cognitive diagnosis model (LCDM) is designed to represent diagnostic classification models as a log-linear model with latent attributes (Henson et al., 2009; J. Templin, Henson, et al., 2010). The LCDM models the conditional probability that examinees in cth latent class with attribute profile $\boldsymbol{\alpha}_c$ obtain a correct response to ith item as follows:

$$\pi_{i,c} = P(X_{i,c} = 1|\boldsymbol{\alpha}_c) = \frac{\exp(\lambda_{i,0} + \boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha}_c, \boldsymbol{q}_i))}{1 + \exp(\lambda_{i,0} + \boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha}_c, \boldsymbol{q}_i))} \tag{2.3}$$

which could also be represented through a logit form:

$$\log\left(\frac{P(X_{i,c} = 1|\boldsymbol{\alpha}_c)}{1 - P(X_{i,c} = 1|\boldsymbol{\alpha}_c)}\right) = \lambda_{i,0} + \boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha}_c, \boldsymbol{q}_i) \tag{2.4}$$

In the logit equation, the linear combination on the right side is also called the kernel. In a kernel, $\boldsymbol{q}_i = q_{i,a}$ is the q-vector for ith item. The intercept, $\lambda_{i,0}$, represents the logit of a response from an examinee whose status of attributes mastery is $\boldsymbol{\alpha}_c = 0$. $\boldsymbol{\lambda}_i$ is the vectorial representation of the coefficients of main

and interaction effects, and $\boldsymbol{h}(\boldsymbol{\alpha}_c, \boldsymbol{q}_i)$ is a vector which contains all types of linear combination between cth latent class attribute profile $\boldsymbol{\alpha}_c$ and ith item's Q-matrix entries $\boldsymbol{q}_i$. The mathematical function of $\boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha}_c, \boldsymbol{q}_i)$ is as follows:

$$\boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha}_c, \boldsymbol{q}_i) = \sum_{a=1}^{A} \lambda_{i,1,(a)} \alpha_{c,a} q_{i,a} + \sum_{a=1}^{A} \sum_{a'>a}^{A} \lambda_{i,2,(a,a')c,a} \alpha_{c,a'} q_{i,a} q_{i,a'} + \cdots \tag{2.5}$$

where $\lambda_{i,1,(a)}$ represents the main effects of attribute a, and $\lambda_{i,2,(a,a')}$ a two-way interaction between ath and a'th attributes.

2.2 Machine Learning

Machine learning is a subfield of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed (Evans & Ossorio, 2018). As described in Chapter 1, generally, machine learning algorithms are categorized as supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning according to different kinds of data and tasks. Supervised learning algorithms are to learn a function that maps an input to an output based on example input-output pairs to produce an inferred function for new datasets. Supervised learning algorithms can be applied for both regression and classification tasks but require the observed input and desired output values or labels (e.g., continuous variables for regression, discrete variables for classification). Supervised learning is widely applied for language modeling, visual object recognition, and making forecasts on businesses data. However, in assessments, the latent variables, including both person and item parameters, cannot be observed directly. In other words, the data collected from the assessments are regarded as unlabeled in machine learning field. This is the reason why a well-defined model with theoretical reasons are relied on to fit the observed data in psychometrics, and very few research studies have used supervised learning for psychometric applications (Cui et al., 2016; Karaca & Hayta, 2016).

2.2.1 Unsupervised Learning and clustering-based psychometric applications

In contrast to supervised learning algorithms, the unsupervised learning algorithms are used to draw inferences from the inputs without labels and find hidden patterns or classes in datasets. The main methods used for unsuper-

vised learning are categorized into two groups: dimensionality reduction and clustering.

Principle components analysis (PCA; Jolliffe, 2002) is one typical and widely used method for dimensionality reduction. The aim of PCA is to extract the features from high dimensional observations to represent the original content (Abedi et al., 2019). PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, also known as principal subspace, in which the variance of the projected data is maximized (Hotelling, 1933). PCA can also be expressed as the maximum likelihood solution of a probabilistic latent variable model. This reformulation of PCA is called probabilistic PCA (Tipping & Bishop, 1999) and it is related to factor analysis (Basilevsky, 1994). When replacing the scalar products in PCA using a nonlinear kernel, the nonlinear generalization, known as kernel PCA (Schölkopf et al., 1998), can be obtained to do the nonlinear dimensionality reduction.

Rule space method (RSM; Tatsuoka, 2009) firstly estimates a traditional unidimensional IRT model, then classified test takers into one of the attribute profiles using the Mahalanobis distance measure. As an extension of the RSM, the attribute hierarchy method (AHM; Leighton et al., 2004) attempts to use a different probabilistic approach (e.g., neural networks; Gierl, Cui, et al., 2008) to classify test takers by putting the relationship between attributes explicitly in the foreground which is ignored in many applications of RSM. The difference between RSM and AHM is that the IRT model in the RSM is calibrated using the observed response pattern, but the IRT model in the AHM is calibrated using the expected response data. Generally, both RSM and AHM impose weaker requirements on the data structure to arrive at test takers classification, but also yields weaker statistical inference.

The clustering, or cluster analysis, is another common technique of unsupervised learning for statistical data analysis used in many fields. Given a set of data observation, the clustering algorithms are to classify each data into a specific group. The data in the same group should have similar properties and observations in different groups should have highly dissimilar properties. In CDM, some clustering methods were introduced to classify test takers into different latent groups based on their responses to the designed item. Chiu et al. (2009) tested the performance of K-means clustering method (Jain, 2010) taking Euclidean distance as instance and hierarchical agglomerative cluster analysis (HACA or HAC; Rokach and Maimon, 2005) using different linkages (e.g., complete linkage, single linkage, average linkage and centroid linkage). Compared with the DCM-based methods (e.g., DINA, RUM), the results showed that although DCM-based methods always achieved more accurate classifica-

tion when the Q-matrix is correctly specified, the K-means clustering methods are attractive because of its simplicity. In addition to K-means, Brusco et al. (2017) also compared K-median method with latent class model (LCM) to cluster dichotomous data. The experimental results (Brusco et al., 2017) showed that both LCM and K-median methods are flexible and can be adapted easily to accommodate other objectives. Although the accessibility and generalizability of K-median clustering is less than that of LCM, the K-median showed a better classification performance than LCM when the number of latent groups K was known. Cui et al. (2016) used self-organizing map (SOM; Kohonen, 2001) to interpret student performance under CDM. The SOM had an input layer, which was composed of large number of input nodes, and an output layer, which was composed of small number of output nodes. Each observed response pattern was firstly put into one input node and then classified into one output node, the one in the closest proximity to it. In their research, the Euclidean distance was used to measure the distance between two vectors. The simulated study based on the framework of DINA showed that the SOM and DINA models are mostly comparable on classification accuracy and consistency for tests with high discriminating items and the advantages of SOM approach become apparent when the model-data misfits are present. Chen and Li (2019) introduced spectral clustering method to do exploratory cluster analysis of items for large-scale assessments supported by a large item pool. The spectral clustering algorithm can deal with the responses from an assessment contains a large number of items (i.e., high dimensional responses) and handle missing data by extracting item clusters via a graphical structure which characterized the similarity structure among items.

Generally speaking, the unsupervised learning methods showed their advantages in simplicity, computational efficiency and handling noises (e.g., missing data, model-data mis-fitting) for psychometric applications. However, in contrast to the theoretic psychometric models, the unsupervised learning methods cannot yield comparable classification results when the appropriate theoretic psychometric models are known. Another disadvantage of unsupervised learning method is that some further data analysis approaches are required to label the clusters. For example, although cluster analysis can place test takers into different latent groups, post hoc techniques are required to discern the attributes from these latent groups.

2.2.2 Semi-supervised Learning

In machine learning field, semi-supervised learning (X. J. Zhu, 2005) concerns with the study of how computers and natural systems learn in the presence

of both labeled and unlabeled data, and it is somewhere between supervised learning (with completely labeled training data; e.g., regression, classification) and unsupervised learning (without any labeled training data; e.g., clustering, dimensional reduction). The research goal of semi-supervised learning is to understand how combining labeled and unlabeled data change the learning behavior, and design algorithms that take advantage of such a combination (X. Zhu & Goldberg, 2009). In a wide range of applications, such as image search (Fergus et al., 2009), natural language parsing (Liang, 2005), and speech analysis (Y. Liu & Kirchhoff, 2014), the semi-supervised learning attracts a great interest because the labeled data is scarce or expensive.

To handle the incomplete labels, the simplest algorithm for semi-supervised learning is based on a self-training (Haffari & Sarkar, 2012; Rosenberg et al., 2005) scheme using bootstrapping with additional labeled data obtained from its own highly confident prediction. Self-training is a wrapper method for semi-supervised learning. This process firstly builds an initial classifier using the correctly labelled examples, and then iteratively classified unlabeled/mislabeled examples, updating the rules for the classifier using the expanded training data, and repeating these steps until some termination condition is reached. The assumption of self-training is that its own prediction tends to be correct. The major advantages of self-training are its simplicity. It works well for the case when the data come from well-separated clusters. However, these methods are heuristic and prone to early error that can reinforce itself by generating incorrectly labeled data. Re-training with this data will lead to an even worse predictor in the next iteration.

Co-Training (Nigam & Ghani, 2000) methods use a pair of classifiers with separate views of the data to iteratively learn and generate additional training labels. When doing diagnostic classifying to a single response pattern, DINA model and DINO model could be viewed as a pair of classifiers with separate views because DINA model assumes that one attribute required by each item cannot be compensated by other required attributes but DINO model assumes that the required attributes can be compensated by others. Like self-training scheme, Co-Training is a wrapper method and widely applicable to many tasks. Co-Training bears strong resemblance to self-training scheme because each classifier uses its most confident predictions on unlabeled instances to teach itself. Two classifiers operate on different views of an observation and the success of Co-Training depends on the following two assumptions (X. Zhu & Goldberg, 2009): 1) each view alone is sufficient to make good classifications, given enough labeled data; 2) the two views are conditionally independent given the class la-

bel. However, practically, it is difficult to find tasks in practice that completely satisfy the conditional independence assumption.

There are some other semi-supervised learning models. Graph-based semi-supervised learning (Fergus et al., 2009) models are applied though efficient spectral methods requiring eigen-analysis of the graph Laplacian and have been shown to be one of the most effective approaches for classification tasks from a wide range of domains; transudative SVMs (Joachims, 1999) extend SVMs with the aim of max-margin classification while ensuring that there are as few unlabeled observations near the margin as possible.

More recently, the techniques to solve the training using noisy labels using artificial neural networks have begun to receive attention. To improve labeling of aerial images, Mnih and Hinton (2012) also developed the deep neural network with a robust loss function to handle label-omission and registration error (or label-error). Larochelle and Bengio (2008) developed Restricted Boltzmann Machine (RBM) for classification that uses a hybrid generative and discriminative training objective. In other words, the RBM classifier learns how to explain observations and how to make predictions simultaneously. Deep Boltzmann Machine (DBM; Salakhutdinov and Hinton, 2009) can also be trained in a semi-supervised manner with labels connected to the top layer. Multi-prediction DBM training (I. J. Goodfellow et al., 2013) and Generative Stochastic Networks (Bengio et al., 2014) improved the performance. In addition, following the idea of minimum entropy regularization (Grandvalet & Bengio, 2005), which performs semi-supervised learning training on unlabeled examples without a generative model, D.-H. Lee (2013) proposed generating "pseudo-labels" as training target for unlabeled data and showed improved performance on MNIST (Modified National Institute of Standards and Technology database) digits recognition with few labeled examples.

2.3 Artificial Neural Networks and Deep Learning

During the last decade, considering the rapidly increasing of data size and development of computational power, research studies and applications using deep learning were growing rapidly. Especially in computer vision, nature language processing and speech recognition (I. Goodfellow et al., 2016; LeCun et al., 2015) methods based on deep learning achieved impressive performances. Deep learning, which is also known as multi-layer artificial neural network, is a specific new subfield of machine learning. The goal of deep learning is to learn representations from data by putting emphasis on learning successive layers

of increasingly meaningful representation (Chollet & Allaire, 2018). In deep learning, the layered representation is learned via models called Artificial Neural Networks (ANNs). The original ANN, artificial neuron or perceptron (Aiserman et al., 1964), was developed to solve problem as the same way of a human brain over 70 years ago. Due to some technology limits, there were not many real applications using ANNs in artificial intelligence (AI) area. Most of the research focused on creating computational models and optimization methods for ANNs. Various types of networks have already been developed according to various types of data (e.g., speech signal, image) and tasks (e.g., classification, prediction, clustering).

2.3.1 Related Concepts of ANNs and Deep Learning

An ANN is a computational system inspired by biological neural systems for information processing in animals' brains. Figure 2.1 shows a biological neuron model and its signal flow from input at dendrites to output at axon terminals. Like a biological neuron model, an ANN is a network of artificial neuron or perceptron (shown in Figure 2.2) and can be characterized as L layers of multiple perceptron (or cells). The lth layer is typically arranged and notated as a vector $\boldsymbol{h}_l = [h_{l1}, h_{l2}, \ldots, h_{lk_l}]$. Layer $\boldsymbol{h}_l$ receives input from the previous layer $\boldsymbol{h}_{l-1}$ and passes its output to the next layer h_{l+1}. The observation vector $\boldsymbol{x}$ is identified as the ANN's first layer h_0 and the output last layer $\boldsymbol{y} = \boldsymbol{h}_L$ is denoted as the output of the ANNs. Figure 2.3 is a simple example of an ANN with 3 perceptrons in the input layer $\boldsymbol{x} = \boldsymbol{h}_0$, one hidden layer $\boldsymbol{h}_1$ with four perceptrons and one output layer $\boldsymbol{y} = \boldsymbol{h}_2$ with two perceptrons. In this example, input of the ANN is a 3-dimensional variable and output of the ANN is a 2-dimensional variable.

In practice, one perceptron h_{li} in layer $\boldsymbol{h}_l$ is calculated using the perceptron in layer $\boldsymbol{h}_{l-1} = \left[h_{(l-1)1}, h_{(l-1)2}, \ldots, h_{(l-1)k_{(l-1)}}\right]$ as input:

$$h_{li} = f_l \left[\sum_{j=1}^{k_{(l-1)}} w_{ji}^{[(l-1)l]} h_{(l-1)j} + b_{li} \right] \tag{2.6}$$

where $w_{ji}^{[(l-1)l]}$ is the connection weight from perceptron $h_{(l-1)j}$ to h_{li}, b_{li} indicates the bias of h_{li}. The calculation in the bracket is a simple linear weighted combination of the perceptron in layer $\boldsymbol{h}_{l-1}$. $f_l(\cdot)$ is the activation function transforms the summed weighted input to an output signal.

Activation functions are important for an ANN to learn and represent complicated and non-linear complex functional mapping between the inputs and

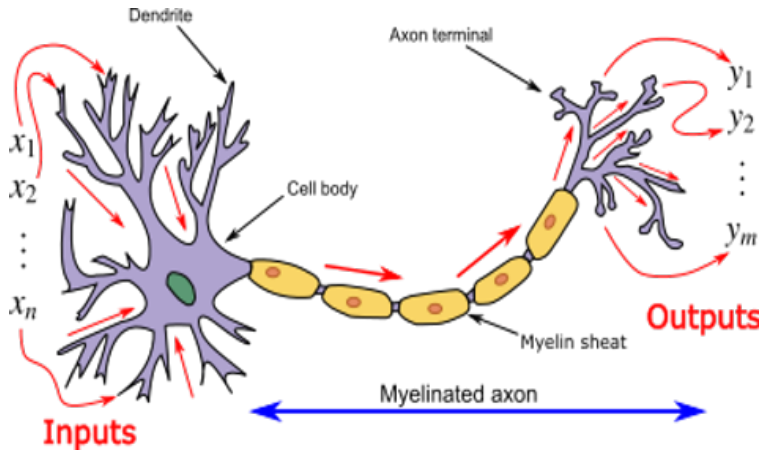

Figure 2.1: Example of a biological neuron.
A biological neuron has dendrites to receive signals, a cell body to process them, and an axon to send signals out to other neurons, the artificial neuron has a number of input channels, a processing stage, and one output that can fan out to multiple other artificial neurons. Source: https://en.wikipedia.org/wiki/Biological_neuron_model

outputs variables. One typical activation function is the logistic function (or sigmoid function in deep learning area):

$$f_l(x) = \frac{\exp(x)}{1 + \exp x} \tag{2.7}$$

which maps real value x to the interval $[0, 1]$, and in deep learning area the sigmoid function is usually used to convert the input to output in a binary classification problem. And the sigmoid function can also be extended to softmax activation function in the multiclass problem as:

$$f_l(x_c) = \frac{\exp(x_c)}{\sum_k^C \exp x_k} \tag{2.8}$$

Hyperbolic Tangent function or *Tanh* is a type of activation function which convert the input to a zero centered output with range between -1 and 1 (i.e., $-1 <$ output < 1) using the following equation:

$$f_l(x_c) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{2.9}$$

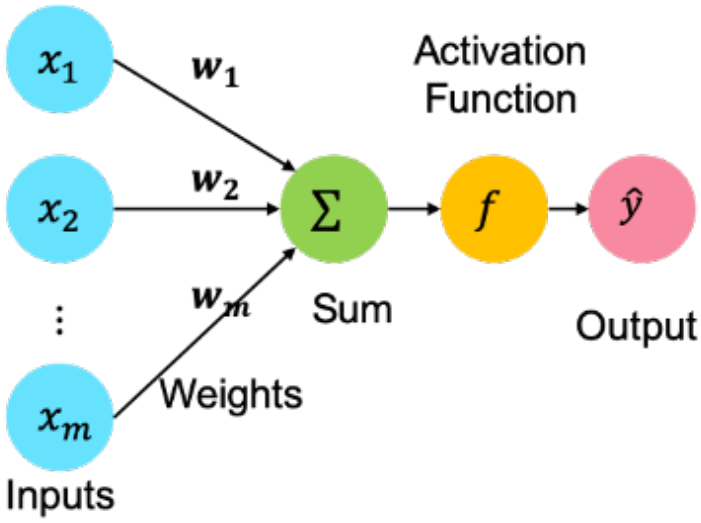

Figure 2.2: Perceptron is the name of a single neuron in deep learning.
It is a linear binary classifier that works for supervised learning to classify the input data. It contains four parts, inputs, weights and bias, net sum and activation function.

However, both *sigmoid function* and *Tanh* suffer from *Vanishing gradient problem*. The problem is that as more layers using certain activation functions are added to ANN, the gradients of the loss function approaches zero, making the network hard to train. To solve the *Vanishing gradient problem*, currently, most deep learning systems use other activation functions, such as Rectified linear unit (ReLU; I. Goodfellow et al., 2016), in the hidden layers between input layer $\boldsymbol{h}_0$ and output layer $\boldsymbol{h}_L$ to avoid a small derivative. The ReLU could be represented as following:

$$f_l(x) = \max(0, x) \tag{2.10}$$

if $x < 0$, $f_l(x) = 0$ and if $x \leq 0$, $f_l(x) = x$. From the mathematical form of ReLU we can see it is very simple and efficient. The only limits of ReLU is that it should only be used with the hidden layer in deep learning systems. For the output layer $\boldsymbol{h}_L$, a *sigmoid function* should be used for a binary classification problem, *softmax function* should be used for a multi-classification problem and a simple *linear function* or *Tanh function* should be used for a regression problem.

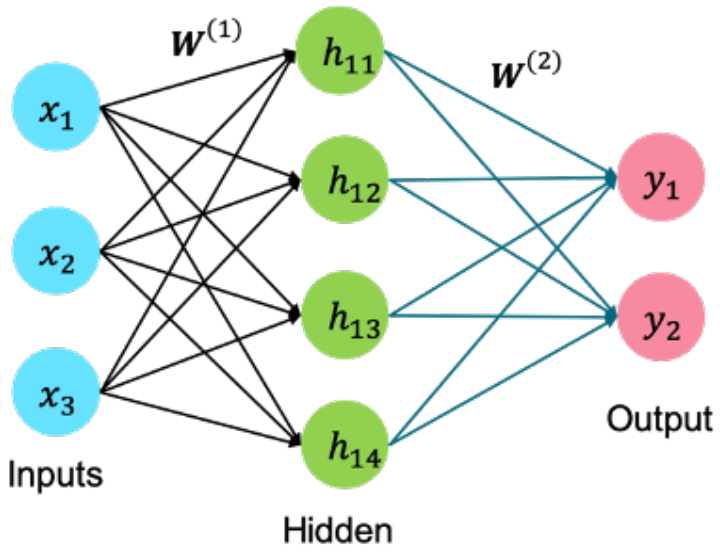

Figure 2.3: A simple artificial neural network with 3 layers.
There are 3 perceptron in the input layer $\boldsymbol{x} = \boldsymbol{h}_0$, one hidden layer $\boldsymbol{h}_1$ with four perceptron and one output layer $\boldsymbol{h}_2$ with two perceptron. In this example, input of the ANN is a 3-dimensional variable and output of the ANN is a 2-dimensional variable.

Backpropagation and Optimization The training of deep learning is implemented as minimizing the difference between target $\boldsymbol{y}$ and deep learning output $\hat{\boldsymbol{y}}(\boldsymbol{x})$ to estimate the connection weights $\boldsymbol{w}$ and the biases $\boldsymbol{b}$. Because the deep learning structure contains multiple layered ANN, the number of unknown parameters is very large and the mathematic representation of y (x) is too complicated to directly use the traditional optimization method. Deep learning widely uses backpropagation (BP; Touretzky and Hinton, 1989; Hinton et al., 2006) method to train and estimate the unknown parameters using different kinds of loss functions and optimization algorithms.

Backpropagation BP computes the gradient of the loss function (will be covered in the following section) with respect to the weights and biases for a single input–output example. Unlike a naive direct computation of the gradient with respect to each unknown parameter individually, BP provides an efficient way to use gradient methods using the chain rule of differentiation through the layers of multilayer ANNs.

In the case of a single hidden layer, we can write the hidden vectors as $\boldsymbol{h_1} = g(\boldsymbol{x}; \boldsymbol{w}^{x1}, b^1)$ which represents $\boldsymbol{h_1}$ as a function of the input layer

$\boldsymbol{h_0} = \boldsymbol{x}$. The output layer is denoted as $\hat{\boldsymbol{y}} = f(\boldsymbol{h_1}; \boldsymbol{w}^{1y}, b^y)$ which represents the predicted output vector $\hat{\boldsymbol{y}}$ as a function of the hidden layer $\boldsymbol{h_1}$. We have the following combined equation that describes the output vector as a function of input vector:

$$\hat{\boldsymbol{y}} = f(g(\boldsymbol{x}; \boldsymbol{w}^{x1}, b); \boldsymbol{w}^{1y}, b^y) = f \circ g(\boldsymbol{x}) \tag{2.11}$$

Let the loss function be $J(\boldsymbol{W}, \boldsymbol{b})$, then we can estimate the connection weight vector between hidden layer $\boldsymbol{h_1}$ and output layer $\hat{\boldsymbol{y}}$ the using the gradient decent method as:

$$\frac{\partial J(\boldsymbol{W}, \boldsymbol{b})}{\partial \boldsymbol{w}^{1y}} = \frac{\partial J(\boldsymbol{W}, \boldsymbol{b})}{\partial \hat{\boldsymbol{y}}} \times \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{w}^{1y}} \tag{2.12}$$

and $\boldsymbol{w}^{1y}$ could be updated as $\boldsymbol{w}^{1y} = \boldsymbol{w}^{1y} - \eta \frac{\partial J(\boldsymbol{W}, \boldsymbol{b})}{\partial \boldsymbol{w}^{1y}}$ with a learning rate η. Therefore, the connection weight vector between input layer $\boldsymbol{x}$ and hidden layer $\boldsymbol{h_1}$ can be estimated using the chain rule as:

$$\frac{\partial J(\boldsymbol{W}, \boldsymbol{b})}{\partial \boldsymbol{w}^{x1}} = \frac{\partial J(\boldsymbol{W}, \boldsymbol{b})}{\partial \hat{\boldsymbol{y}}} \times \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{h_1}} \times \frac{\partial \boldsymbol{h_1}}{\boldsymbol{w}^{x1}} \tag{2.13}$$

and $\boldsymbol{w}^{x1} = \boldsymbol{w}^{x1} - \eta \frac{\partial J(\boldsymbol{W}, \boldsymbol{b})}{\partial \boldsymbol{w}^{x1}}$. To estimate biases b^y and b^1, we could use similar training procedure. Training deep learning structure using BP can be accelerated dramatically using modern parallel programming paradigms through multi-core CPU, Graphic Processing Unit (GPU) or Tensor Processing Unit (TPU) when the deep learning contains multiple layers and many perceptron.

Loss function Generally, the loss functions used in deep learning could be divided with regard to the data analyzed. The task to predict discrete variables (e.g., dichotomous and polychotomous responses, latent classes, attribute mastery status) is a classification problem. In machine learning, the method is "one-hot" coding which is also known as dummy coding in psychometrics. A discrete output variable with Kpossible states $y \in \{1, 2, \ldots, K\}$ is recoded as $\boldsymbol{y^*} = [y_1^*, y_2^*, \ldots, y_K^*]$ with $y_j^* = 1$ and $y_{k \neq j}^* = 0$ if $y = j$. For a binary classification problem, the output uses a sigmoid activation function, and a multi-classification problem uses softmax activation function to predict the output. Then the predicted $y_k^* = P(y_k^* = 1 | \boldsymbol{x})$ are the probability of the classification represented by the output neurons given input data $\boldsymbol{x}$. To measure the difference between predicted output $\hat{\boldsymbol{y}}^* = \{\hat{y}_k^*\}$ and the observed output

$\hat{\boldsymbol{y}}^*$, the loss function can be defined as:

$$L = \frac{1}{N}\sum_{n=1}^{N}[-\sum_{k=1}^{K} y_{kn}^* \log \hat{y}_{kn}^*] \tag{2.14}$$

which is referred to as cross-entropy (I. Goodfellow et al., 2016; Murphy, 2012) as this function relates to the measure of entropy $H(\boldsymbol{p}, \boldsymbol{q}) = -\sum_i p_i \log q_i$ in information theory. N is number of observations.

For continuous variables, such as the latent ability of students in IRT model, the task of prediction is a regression problem. Training is implemented as minimizing the mean squared error (MSE) between the observed output y and predicted output $\hat{\boldsymbol{y}}$ as following:

$$MSE = \arg\min\left\{\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n)^2\right\} \tag{2.15}$$

Typical Deep Learning Architectures In the field of deep learning, during the past decade, a large number of deep neural networks (DNNs) with different architectures, algorithms and names were designed to solve various tasks. ¬Here, four basic types of the deep learning architectures will be discussed, and they are: Autoencoder (AE), Deep Feedforward Network (DFN), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN).

Autoencoder (AE) Autoencoder (AE; Liou et al., 2008) is a type of neural network which focuses on unsupervised learning and extracting latent information (or features) from the observation. The goal of an AE is to compress data into a lower dimensional feature/code and reconstruct outputs that closely matches the original input from that code. Architecturally, an autoencoder has an input layer, an output layer and one or more hidden layers connecting them. The input layer and output layer have the same number of artificial neurons. Generally, an autoencoder always consists of two parts, the encoder and the decoder. Encoder is to find the low dimensional code and decoder is to decompress that code into reconstructed outputs. Both of the two parts can be defined as transitions Φ and Ψ mathematically:

$$\begin{aligned} \Phi &: \boldsymbol{X} \to \mathcal{F} \\ \Psi &: \mathcal{F} \to \boldsymbol{X}' \end{aligned} \tag{2.16}$$

where Φ indicates a dimensional reduction from high dimensional observation space $\boldsymbol{X}$ to a low dimensional feature/code space $\mathcal{F}$ by removing the noises or

redundant information, and Ψ indicates a reconstruction process to output $\boldsymbol{X}' = \Psi \circ \Phi(\boldsymbol{X})$ which has the same dimension as $\boldsymbol{X}$, the autoencoder is trained via BP by minimizing the target function as following:

$$\Phi, \Psi = \arg\min ||\boldsymbol{X} - \boldsymbol{X}'||^2 \tag{2.17}$$

where the L-2 Regularization or Euclidean distance can be replaced by L-1 Regularization or cross entropy for different kinds of data and measurement. When implementing an autoencoder, there are some "tricky" things to make the autoencoder to learn useful features. The first one is to keep the coder layer small; the second one is to add random noise to the inputs of autoencoder; and the third one is to use a sparsity constraint by using regularization.

Deep Feedforward Network (DFN) Feedforward neural network is an artificial neural network wherein connections between neurons do not form a cycle (Zell, 1994). The feedforward neural network is the first and simplest type of ANN. A perceptron is a kind of feedforward neural network. Deep Feedforward Network (DFN) is a class of ANNs that consists of multiple layers of artificial neurons and represents multilayer network architecture with many hidden layers (Schmidhuber, 2015). In DFN structure, each neuron in one layer has directed connection to the neurons of the subsequent layer. The input layer constitutes the observed input, and the hidden layer characterizing the abstract description of this input. The goal of the output layer is to only perform the prediction or classification. In practice, DFNs are applied for supervised learning and trained through the backpropagation algorithm.

Convolutional Neural Network (CNN) In DFNs, one neuron in one layer are connected with all the neurons of the adjacent layers. This architecture is also known as fully connected deep networks. However, applying a DFN with fully connected structure to high dimensional observation, such as image which may contain millions of pixels, is not practical because such high dimensional inputs requires huge number of parameters to be optimized. Inspired by biological processes of the visual cortex (Matsugu et al., 2003), Convolutional Neural Networks (CNNs) bring a solution by applying a convolution operation to the input. A typical CNN consists of three kinds of layers: convolutional layer, pooling layer and dense layer (or full connected layer).

Taking 2D image as input, the convolutional layer can be viewed as converting a current pixel's value to a linear combination of itself and its neighbors within a specific distance. In other words, the convolution operation is to

extract more abstract information and remove high-frequency noise through imitating the receptive field of a single sensory neuron of human vision.

The pooling layers (subsampling layers) in the CNN architecture are used to subsample the feature maps by combining the outputs of neuron clusters at its prior layer (convolutional layer) into a single neuron in the pooling layer. There are two common pooling methods, max pooling and average pooling. Max pooling uses the maximum value from each cluster of neurons at the prior layer. Average pooling uses the average value from each cluster of neurons at the prior layer.

The full connected layer is to obtain high-level reasoning after several convolutional and pooling layers. Neurons in a fully connected layer have connects to all neurons in the adjacent layers. The full connected layers are to perform some specific tasks (e.g., classify, predict) on the features extracted by the convolutional and pooling layers.

As the founder of CNN and deep learning pioneer, LeCun et al. (1999) developed one of the very first successful applications of CNNs, named LeNet5, to classify hand-written numbers. Currently, some other common CNN applications are *AlexNet* (Krizhevsky et al., 2012), *ZF Net* (Zeiler & Fergus, 2014), *GoogLeNet* (Szegedy et al., 2015), and *ResNet* (He et al., 2016).

Recurrent Neural Network (RNN) RNNs (Graves et al., 2013) are a type of ANNs that turned out to be particularly suitable for modeling sequence data such as language, text, music, or finical data. In contrast to the previous ANN types, RNNs has two advantages: first one is that RNNs can accumulate the previous information of input from each time step in sequence data into a hidden state to model the problems within time; the second advantage is that RNNs has the ability to learn representations from sequences with variable length (e.g., sentences, documents, item stem).

In RNNs, the input for the next state $t+1$ (i.e., the output of the current state o_t or h_t) consists of the observation x_t from state t and the hidden state h_t. Therefore, the current hidden state h_t has summarized the information from the current observation x_t and the previous historical information in h_{t-1}:

$$o_t = h_t = f(x_t, h_{t-1}) = f(\boldsymbol{U}x_t + \boldsymbol{W}h_{t-1}) \tag{2.18}$$

with $h_{t-1} = g(x_{t-2}, x_{t-3}, \ldots, x_0)$. $\boldsymbol{U}$ and $\boldsymbol{W}$ are weight matrices. In RNNs, sigmoid function, tanh function or ReLU are often used as the activation function f. The state value of the hidden layer in turn can be viewed as a function of all previous observed values $x_{t-2}, x_{t-3}, \ldots, x_0$. This essentially produces a distribution of the next observation given all previous observations as follow-

ing (von Davier, 2018):

$$p(x_0, \ldots, x_t) = p(x_0) \prod_{i=1}^{t} p(x_i | x_{i-1}, \ldots, x_0) \tag{2.19}$$

In contrast to Hidden Markov Model (HMM; Beal et al., 2002) which also contains hidden states, RNNs do not hold the Markov assumption. HMM is much simpler than RNN and relies on strong assumptions. The Markov assumption makes HMM be efficient on learning and inference. RNN is more complicated but more powerful. The distributed hidden state that allows RNN to store a lot of information about the past, and the artificial neural network allows them to update their hidden state in complicated ways.

Strengths and Challenges to the Use of Deep Learning in Psychometrics

Deep learning is largely responsible for the growth in the use of AI. This technology has given computer programs the ability to complete some tasks, such as recognizing speech and human face detection, as good as a human being. It is now being used to guide and enhance all sorts of key processes in medicine, finance and marketing. ANNs (Gierl, Cui, et al., 2008; Gierl, Wang, et al., 2008) have also been proposed as an attractive approach to convert a response pattern into defensible latent variables (e.g., latent trait, latent class). In this session, we will discuss the strengths and advantages that have earned deep learning the popular status as well as current challenges that need to be addressed in psychometrics.

Strengths of deep learning

Feature hierarchy As one pioneer of deep learning, Bengio gave a definition of deep learning: "Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features (Bengio, 2012, p. 1)" From his definition, we know that one advantage of deep learning is to represent the input distribution at multiple levels, which is also known as feature hierarchy or hierarchical feature representation. Figure 2.4 shows an example of feature hierarchy in the human face recognition tasks (H. Lee et al., 2009). With those face images as input, deep learning is able to represent brightness contrast in the first hidden layer. The brightness contrast represented some common image features, such as corner, edges or special texture. From that, it matches the contrast with known objects

like eyes or nose as the output of the hidden layer 2. In the hidden layer 3, deep learning could extract more abstractive human face in terms of the facial organs learned in the hidden layer 2. Generally speaking, the feature hierarchy of deep learning is to use layers of data representation to make sense of seemingly unrelated data. In psychometrics, the deep learning with an appropriate architecture is also expected to output different latent variables at different hidden layers from the observed response patterns. Like the example of human face detection in Figure 2.4, the first layer (or input layer) of the deep learning architecture is the observed response patterns; and the second layer derives lower level patterns/features from the observation; then the third layer uses these lower level features to gradually identify higher level features, through many layers, hierarchically.

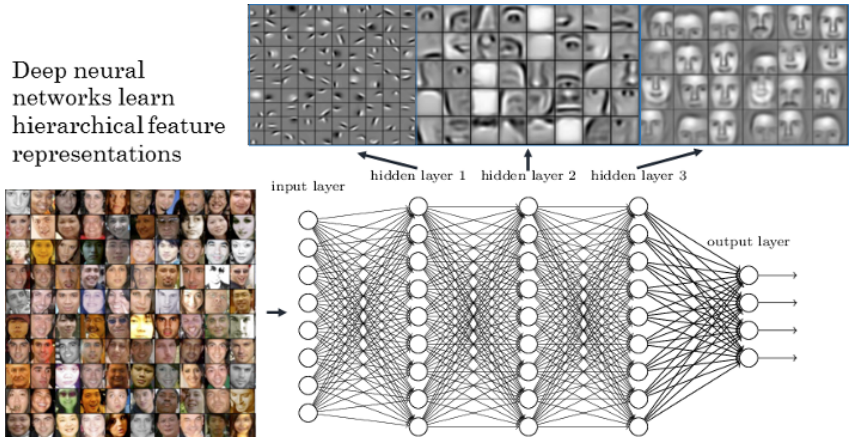

Figure 2.4: An example of feature hierarchy in the human face recognition tasks. An example of feature hierarchy in the human face recognition tasks (H. Lee et al., 2009). With those face images as input, deep learning is able to represent brightness contrast in the first hidden layer. The brightness contrast represented some common image features, such as corner, edges or special texture. From that, it matches the contrast with known objects like eyes or nose as the output of the hidden layer 2. In the hidden layer 3, deep learning could extract more abstractive human face in terms of the facial organs learned in the hidden layer 2. Source: https://medium.com/@leishi_51564/deep-learning-619fe06a3fd8

General approximation In mathematical theory of ANNs, the universal approximation theory states that a feed-forward depth-2 network, which contains a single hidden layer, with a finite number of neurons and suitable activation function can approximate any continuous function on a compact domain to any desired accuracy (Csáji et al., 2001; Hornik et al., 1994). In these

classical research work of universal approximation theory, the sigmoid function was chosen as the activation function (Barron, 1994; Cybenko, 1989). Since the efficiency of computation, ReLU networks are used more and more in current deep learning research. Lu et al. (2017) showed a universal approximation theorem for width-bounded ReLU networks: width-$(d+4)$ ReLU networks, where d is the input dimension, are universal approximators. Hanin (2019) also proved that ReLU nets with width-$(d+1)$ can approximate any continuous convex function of d variables arbitrarily well. Their results also gave quantitative depth estimates for the rate of approximation of any continuous scalar function on the d-dimenstional cube $[0,1]^d$ by ReLU nets with width-$(d+3)$. The current research work stated that the width and the depth are two key components in the design of a deep learning architecture. In summary, deep learning provides information-theoretically optimal approximation of a very wide range of functions and function classes used in mathematical signal processing (Grohs et al., 2019).

No need for feature engineering As a fundamental job in machine learning to improve model accuracy, feature engineering is the process to extract features from raw data to provide better representation of the underlying problem. The process sometimes requires domain knowledge about a given problem. For example, the knowledge of the automobile industry when working with the relevant data can be used to specify two features Horsepower (HP) and revolutions per minute (RPM) to then to create an additional feature like Torque from the formula $\text{TORQUE} = \text{HP} \times 5252/\text{RPM}$. One advantage of deep learning's main advantages over other machine learning algorithms is its capacity to execute feature engineering without relying on domain knowledge. From the data, deep learning algorithm can explore the features that correlate and combine them to achieve prediction or classification tasks without being explicitly programmed. This ability means that sometimes deep learning algorithm can save months of work and the features extracted are more robust than the outputs of manual feature engineering especially when these features are new or more complex that human might miss.

Challenges for Psychometrics As a new research area, there are some challenges that need to be addressed in deep learning field. The first challenge is that training deep learning algorithms requires a large data set. The reason of large data requirement is that deep learning needs to learn about the domain and then solve the problem. When the training of deep learning begins, the algorithm starts from scratch. In contrast to other machine learning algorithm

requiring domain knowledge, the deep learning algorithm needs a huge number of parameters to tune to achieve the tasks in a domain. For example, Deep Patient (Miotto et al., 2016) was a deep learning program that was applied to patient records of more than $700,000$ individuals at Mount Sinai Hospital in New York. After a long training period, Deep Patient was able to detect certain illnesses better than human doctors.

The second challenge is that the computation of deep learning is known as a "Black Box" and the output of deep learning is uninterpretable. Although deep learning-based AI systems obtained impressive achievement in some domains, such as Alpha Go and its successors, how these systems achieved their outputs is hard to be explained like other machine learning algorithms (e.g., support vector machine, tree-based methods). Given the aim of successful prediction or classification in supervised learning, the black box nature of deep learning is not problematic. For good practice, the deep learning applications involve data with correct labels that can be split into three sets for training, validation and testing data. The training set is used to train the parameters in the deep learning architecture; the validation set is used to examine the parameter estimation of the deep learning to avoid over-fitting; and the testing set is used to evaluate the prediction accuracy of the deep learning.

In contrast, because psychometrics focuses on exploring the underlying distribution (i.e., latent variables) by analyzing the observed response data, it is not possible to evaluate a deep learning model just using empirical data as is common in deep learning applications. Cui et al. (2016) suggested the potential exploratory use of an unsupervised ANN known as a "self-organizing map" (SOM; Kohonen, 2001) in CDM and training a multiple layered perceptron (MLP) using simulated data based on expected response patterns. Briggs and Circi (2017) also suggested to gather data with the most common observed student response patterns and then have content experts to give each pattern a holistic score with respect to each attribute of interest regarding to the Q-matrix. Compared with the expected response patterns, the uncertainty of deep learning training might be reduced because the training set would more closely resemble the type of data for which estimates of attribute probabilities are desired in theoretical psychometric models.

2.4 The General Idea of Deep Learning-based Computational Psychometric Methods

This research is to explore the feasibility of using deep learning to extract latent person variables (e.g., latent trait, latent class) concerned by psychomet-

rics. In contrast to the previous research work using ANNs (e.g., Briggs and Circi, 2017; Cui et al., 2016; Paulsen, 2019), this research is to propose a general deep learning-based method, which is named as Deep Learning-based Computational Psychometric Methods (DLCPMs). As shown in Figure 2.5, the proposed family of DLCPMs generally consist of three parts: *observed inputs*, *feature extraction* (or *latent variable extraction*), and targeting.

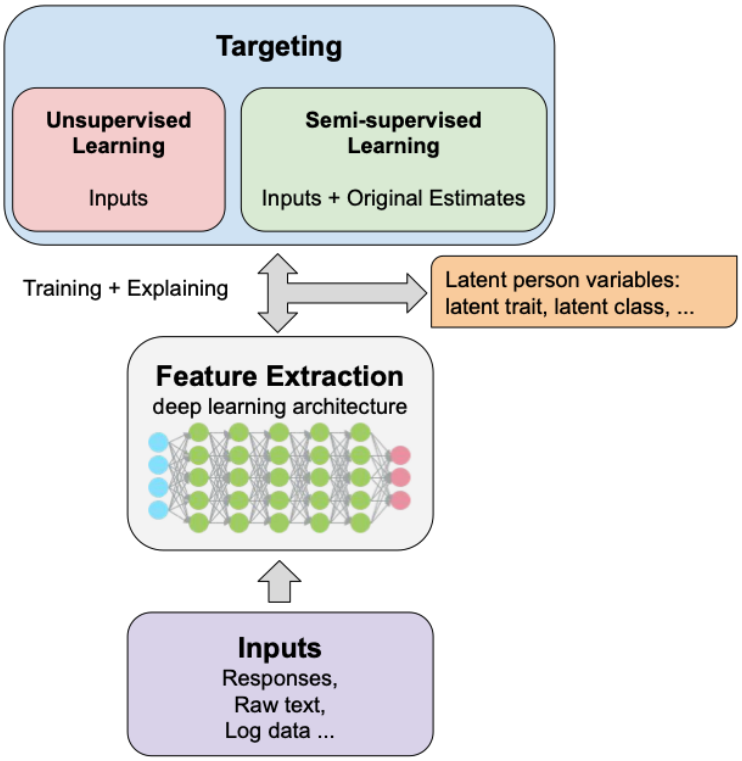

Figure 2.5: The diagram of the proposed family of Deep Learning-based Computational Psychometric Models (DLCPMs).
DLCPMs generally consist of three parts: observed inputs, feature extraction and targeting.

Observed inputs indicates the observed data such as students' responses to items in a designed assessment, or interactions within an online learning environment such as raw text, log data and sequential data. In this research, because we are focusing on exploring the feasibility of using deep learning to determine the latent person variables that psychometric models are concerned with, the inputs are the students' dichotomous responses to items.

Feature extraction or *latent variable extraction* is to convert the observed response pattern to latent person variables that psychometric models are concerned with using deep learning techniques. For IRT, this latent variable refers

to students' ability (continuous variables) and for CDM the latent variable is attribute profiles (discrete variables). Due to feature hierarchy of deep learning, the structures of the deep learning architecture differed according to different types of latent variables we wanted to exploit from the inputs.

To consider both prediction on a new dataset and explaining on current training dataset simultaneously, the *targeting* is two-folded: training and explaining. Like typical machine learning tasks, process of training is to estimate appropriate parameters contained in the deep learning architecture for future prediction on new dataset; process of explaining is to make the latent variables exploited by the deep learning-based feature extraction have the same interpretation as the theoretic psychometric models. According to different research goals, the targeting could be modified for unsupervised learning for data exploration (e.g., determining Q-matrix or the hierarchical structure of attributes) from the raw data, or for semi-supervised learning for obtaining more robust latent variable estimation when there is noise contained in the data (e.g., inaccurate Q-matrix, large amount of non-ignorable missing values).

In this research, there are two applications of using DLCPMs for cognitive diagnostic modeling, the first one is an unsupervised deep clustering method for CDM and Q-matrix reconstruction; the second one is a semi-supervised learning Co-Training method for CDM. The detailed methods of these two applications will be described respectively in Chapter 3 and Chapter 4.

Chapter 3

An Unsupervised Learning Artificial Neural Network for Cognitive Diagnostic Measurement

3.1 Introduction

The purpose of cognitive diagnostic modeling (CDM) or diagnostic measurement is to provide students' knowledge mastery states based on their responses to items from carefully designed assessments. Because of the ability to provide educators diagnostic feedback from students' assessment results, CDM has been the focus of much research in the last decade. Various types of diagnostic classification models (DCMs); such as the deterministic inputs, noisy "and" gate (DINA; Junker and Sijtsma, 2001) model, the reparametrized unified model/fusion model (RUM; Hartz, 2002), and the log-linear cognitive diagnosis model (LCDM; Henson et al., 2009); are designed based on different cognitive theories or assumptions about the relationship between item response patterns and attribute patterns, or based on the item response function.

Although different DCMs have various item response functions, all recent DCMs can be categorized into two groups. The first group of DCMs is built with a fully probabilistic model (parametric) structure and rely on latent variables (e.g., item parameters, guessing parameter, slipping parameter), such as the DINA, DINO, RUM and LCDM (A. Rupp et al., 2010). When given a specific DCM and response data, current methods use maximum likelihood estimation

(MLE; Chiu et al., 2016; Junker and Sijtsma, 2001) or Bayesian techniques including Markov chain Monte Carlo (MCMC) methods (Henson et al., 2009) to estimate both item and person parameters by fitting the given DCMs. To classify the examinees into one of several latent groups, the second group of DCMs is designed based on classification without requiring fully probabilistic models. These classification-based approaches include Rule-Space Methodology (RSM; Tatsuoka, 2009), the attribute Hierarchy Method (AHM; Gierl, Cui, et al., 2008), and cluster analysis (Brusco et al., 2017; Chiu et al., 2009).

A Q-matrix indicates the relationship between items and attributes in an assessment. Q-matrices are often carefully designed by assessment experts. Some existing research and their experimental results, however, have shown that Q-matrices constructed by content experts do not always reflect the relationship precisely and may require empirically-driven modifications (Bradshaw et al., 2014; Tjoe & de la Torre, 2014). Items within an assessment may vary with respect to their diagnostic quality, or the discriminating power of the item to determine the success of the diagnosis. An item with high discrimination is one on which students who have mastered the attributes required by the item are expected to have a high probability of responding to the item correctly, while students who have not are expected to have a low probability. Items with low discriminating power contribute less statistical information to the estimate of student attribute mastery. In the previous research studies, the performances of all DCMs are sensitive to the diagnostic quality of items and the accuracy of Q-matrices (Kunina-Habenicht et al., 2012; R. Liu et al., 2017).

Due to the impressive achievement of deep learning and Artificial Neural Networks (ANNs) in other research fields (e.g., natural language processing, computer vision), ANNs have been proposed as an attractive approach in some research studies (Cui et al., 2017; Cui et al., 2016; Paulsen, 2019) to convert a pattern of item responses into a diagnostic classification, or attribute profiles. As shown in Figure 3.1, an ANN is a computational system inspired by biological neural systems for information processing in animals' brains. An ANN is built on inputs being translated to outputs through a series of neuron layers. It consists of three types of layers: an input layer, a hidden layer(s), and an output layer. Each layer consists of some neurons (or nodes), which are connected to the nodes in the next layer. The value of a node is calculated using the values of nodes in the previous layer. Each layer, except for the input layer, uses the output of its previous layer as the input.

Both supervised learning ANNs and unsupervised learning ANNs were applied in some CDM research studies (Cui et al., 2017; Cui et al., 2016; Paulsen, 2019). To train the supervised learning ANNs, the ideal response patterns

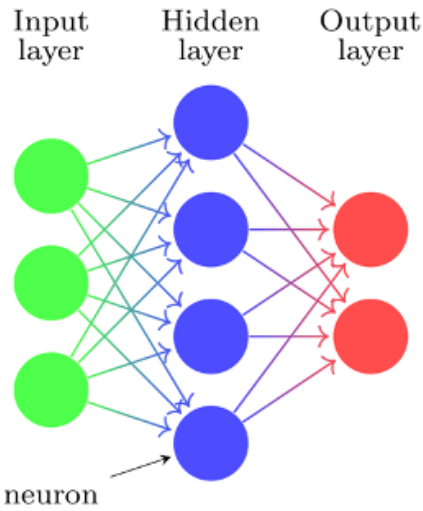

Figure 3.1: Example of a simple artificial neural network (ANN).
A basic ANN architecture is formed by one input layer, one hidden layer, and one output layer. Each circle indicates the artificial neuron or node, and an arrow indicates the weights between two nodes.

were required to be set as the input layer and the associated attribute profiles as the output layer. Cui et al. (2016) and Paulsen (2019) hypothesized the DINA model with both slipping and guessing equalling to 0 to synthesize ideal responses to train a multilayer perceptron (MLP). The experimental results showed that the classification accuracy of the supervised learning ANNs was not appreciated even in the simulated study. A disadvantage of applying supervised learning ANNs for CDM is the difficulty in determining how to create the ideal response patterns using a DCM because both the DCM and its parameter values are difficult to hypothesize. In addition to supervised learning ANNs, Cui et al. (2016) used one type of unsupervised learning ANN, self-organizing map (SOM; Kohonen, 2001), to classify test-takers into different latent groups for CDM. One disadvantage of the unsupervised learning ANNs is that some further data analysis approaches are required to label the latent groups, or the clusters. For example, although cluster analysis can place test-takers into different latent groups, post hoc techniques are required to discern the attributes from these latent groups. Additionally, the ANN methods can produce very unstable and unappreciated estimation unless a great deal of care is taken to conduct sensitivity analyses (Briggs & Circi, 2017).

The motivation of this research comes from the following three issues among the existing research studies of CDM: (1) the quality of Q-matrix and the discriminating power of items impacts the diagnostic classification of DCMs; (2) the current supervised leaning ANN methods for CDM requires assumption of a specific, known item response function; (3) the unsupervised learning ANNs method for CDM requires further data analysis to interpret the results. To solve these three issues, we sought to design a general approach to estimate attribute profiles and to reconstruct an inaccurate Q-matrix. The proposed method contains two steps: (1) an unsupervised learning ANN to estimate student attribute profiles only requiring partial Q-matrix information without an assumption of an item response function; (2) Q-matrix reconstruction using the item response data and the attribute estimates.

This paper contains three sections. In the first part, the designed method for unsupervised learning ANN attribute estimation and Q-matrix reconstruction is described. In the second part, a simulation study is conducted to test three test factors' effects on the performance of the designed method and to make a comparison with latent class model-based methods. In the last part, the benefits and challenges of our methodology are summarized, and future research is also outlined.

3.2 Method

The diagram of the proposed method is shown in Figure 3.2. In this research, we firstly designed an unsupervised learning ANN to estimate the attribute profiles of test-takers without hypothesizing the item response function. Secondly, we proposed a Q-matrix reconstruction algorithm to correct or reconstruct the misspecified or missing elements of the Q-matrix (i.e., the complex items' q-vectors) through applying a K-means clustering algorithm.

To successfully apply our method to CDM, the proposed method assumed that three test conditions hold. First, the number of attributes is known. This is a typical assumption of all DCMs. Second, the q-vectors of simple items are correctly specified. This assumption reflects many real-world scenarios where simple structure items are easier for domain experts to specify in comparison to complex structure items. It may not be realistic that the elements for all simple structure items of a Q-matrix are correctly specified, but that is an assumption we will make here. The last assumption is that, for each attribute, at least one simple item with high discrimination measures it, which is a similar but less complex mathematical constraint on the Q-matrix for completeness (Chiu et al., 2009; DeCarlo, 2011).

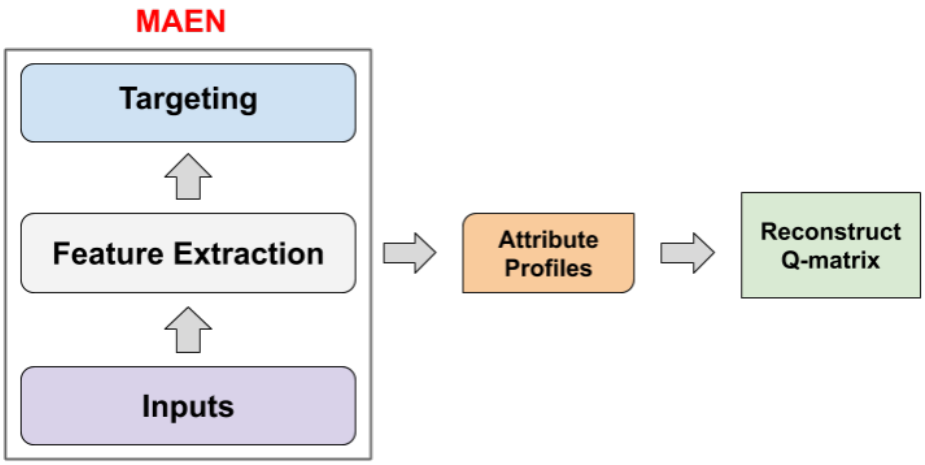

Figure 3.2: Structure of the proposed method.
The proposed method consisted of an unsupervised learning algorithm (modified autoencoder network)

3.2.1 Unsupervised Learning ANN for Attribute Profile Estimation - MEAN

The proposed unsupervised learning ANN was conducted based on an autoencoder, which is an ANN used for unsupervised learning of efficient code/feature extraction (Chiu et al., 2009; DeCarlo, 2011; Liou et al., 2008). The reason for introducing an autoencoder was based on the assumption that when extracting the latent person variables (i.e., the attribute profile for each examinee), the observed responses could be reconstructed using these estimates of latent person variables. An autoencoder (as shown in Figure 3.3) consists of two neural networks, the encoder and the decoder. The widths (i.e., number of nodes) of the input layer and output layer of an autoencoder are the same. The encoder and decoder can be defined as transitions Φ and Ψ:

$$\begin{aligned} \Phi &: \boldsymbol{X} \rightarrow \mathcal{F} \\ \Psi &: \mathcal{F} \rightarrow \boldsymbol{X}' \end{aligned} \tag{3.1}$$

where Φ indicates a dimensional reduction from high dimensional observation space $\boldsymbol{X}$ to a low dimensional feature/code space $\mathcal{F}$ by removing the noises or redundant information, and Ψ indicates a reconstruction process to output $\boldsymbol{X}' = \Psi \circ \Phi(\boldsymbol{X})$ which has the same dimension as $\boldsymbol{X}$, the autoencoder is

trained via BP by minimizing the target function as following:

$$\Phi, \Psi = \arg\min ||\boldsymbol{X} - \boldsymbol{X}'||^2 = \arg\min ||\boldsymbol{X} - \Psi \circ \Phi(\boldsymbol{X})||^2 \qquad (3.2)$$

where the L-2 Regularization or Euclidean distance can be replaced by L-1 Regularization or cross entropy for different kinds of data and measurement. When implementing an autoencoder, there are some "tricky" things to make the autoencoder to learn useful features. The first one is to keep the coder layer small; the second one is to add random noise to the inputs of autoencoder; and the third one is to use a sparsity constraint by using regularization.

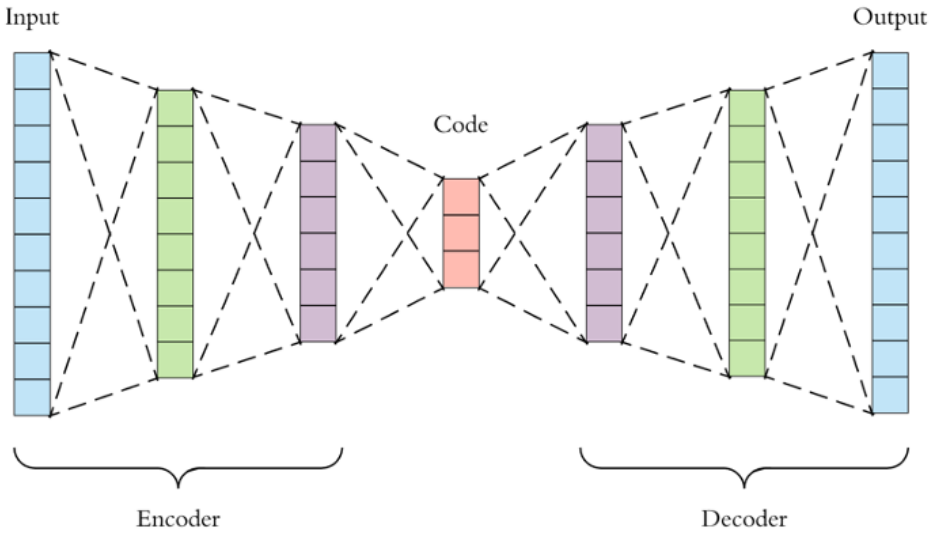

Figure 3.3: Example of Autoencoder
An autoencoder consists of two networks: encoder neural network from the input layer to the code layer, and decoder neural network from the code layer to the output layer. The encoder is to compress the observation to code, but the decoder is to reconstruct the output from code. Source: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

As shown in Figure 3.4, the proposed method, a modified autoencoder network (MAEN), consisted of three parts: observed input, feature extraction, and targeting.

Observed Input The observed input is the input layer of the MAEN, and each node indicates a student's dichotomous response to one item. The width (i.e., number of nodes) of the input layer is equal to the number of items con-

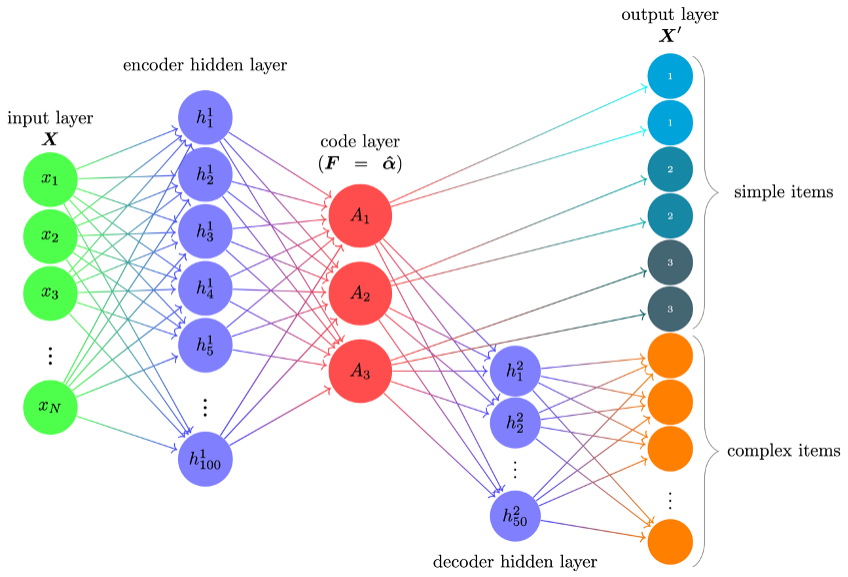

Figure 3.4: Structure of the proposed modified autoencoder network (MAEN) This is the structure of the propose modified autoencoder network for estimating attribute profiles.

tained in the assessment. For example, if an assessment consisted of 20 items, the width of the input layer equaled 20.

Feature Extraction The feature extraction, which is also the encoder of the MAEN, had a similar function as the encoder of a typical autoencoder. The target of feature extraction is to convert the observed input to the attribute profile for each test-taker. The feature extraction contained two hidden layers, an encoder hidden layer and a code layer. We used the Rectified Linear Unit function (ReLU; Nair and Hinton, 2010) as the activation function for the first two hidden layers, and the sigmoid function as the activation function for the code layer. The reasons of using ReLU in the first two hidden layers are (1) ReLU function is very simple; (2) using ReLU can fix the vanishing gradients problem (Ide & Kurita, 2017). Using sigmoid function as the activation function of the output layer is that sigmoid function is to compress the inputs into the 0 to 1 range which indicates the probability of correct response to items.

The output of the code layer was the estimation of attribute profile, and the width of the code layer is equal to the number of attributes measured in the assessment. For example, when analyzing data from an assessment that

measures three attributes, the width of the code layer is 3. The widths of the encoder hidden layers were set as 100 because of the following two reasons. The first reason was the approximation theory of ANN has indicated that width-bounded ReLU networks, at least width-$(d + 4)$ ReLU networks, where d is the observation dimension, are universal approximators hanin2019universal, lu2017expressive. In the training procedure, we set the dropping out ratio equals to .3 to avoid over-training and to improve the generalization of artificial neural networks. The dropping out ration is the percentage of nodes that are randomly selected to be deactivated in the training procedure. Some research studies (Dahl et al., 2013) show that dropping out in a neural network is a simple and effective regularization method. The first thing needed to be considered in seting the number of hidden nodes is to make sure the number of activated nodes in the hidden layers in the training is greater than width-(d +4). The second thing need to be considered is that the size of observations is relatively small compared with the number of parameters in the ANNs, the width of the hidden layer cannot be too wide. After comparing the 200, 100, and 50 hidden nodes structure in the training, the 100 hidden nodes was chosen for a trade-off. The number of activated nodes (i.e., 70) holds that the width of encoder hidden layer is wider than (d+4), and the number of parameters for each training is 1470 which is less than the number of observations (i.e., 2000) in each training iteration.

Targeting The third part was targeting, which worked as the decoder in a typical autoencoder. It used the output of the code layer as the input to reconstruct the item responses. As shown in Figure 3.4, the targeting consisted of the code layer, one decoder hidden layer, and the output layer. In the previous research studies of using supervised ANNs, the item response functions (i.e., DINA model) were required to be hypothesized. In contrast, the decoder could reconstruct the observed inputs from the coder layer outputs without specifying the item response function, because the mathematical theory of ANNs states that a feed-forward depth-2 network, which contains a single hidden layer, with a finite number of neurons and suitable activation function can approximate any continuous function on a compact domain to any desired accuracy (Csáji et al., 2001; Hornik et al., 1994).

As mentioned before, one disadvantage of the existing unsupervised learning ANNs for CDM is that some further data analysis approaches are required to label the clusters. The reason causing that issue is the full connection between the attribute profiles and the response patterns in those ANN structures. In our method, we overcame this disadvantage by modifying the decoder net-

work by using a sparse connection strategy according to the partially known Q-matrix: (1) for each simple item of which the q-vector is known, the corresponding reconstructed output is connected directly to the code node which corresponds to the attribute this item measures; and (2) for each complex item of which the q-vector is unspecified, the corresponding reconstructed output node is fully connected to all decoder hidden nodes because the relationship between complex and attributes cannot be determined without knowing q-vector. The advantages of using a sparse connection are (1) not requiring strong prior knowledge of the relationship between item and attributes, (2) not causing classification switching compared with the fully connected structure (i.e., each code node and all output nodes are connected), and (3) providing the interpretable coder layer outputs.

Let $\mathcal{H}_h^d$ and x_i' denote the hth node in the decoder hidden layer and the ith node in the output layer, respectively. This strategy can be mathematically represented as follows:

$$\begin{aligned} x_i' &= p(x = 1|\mathcal{F}) \\ &= \begin{cases} \sigma(w_{a,i}\mathcal{F}_a + b_i) & i\text{th item only measure } a\text{th attribute} \\ \sigma(\sum_{h=1}^{50} w_{h,i}\mathcal{H}_h^d + b_i) & i\text{th item is complex item} \end{cases} \end{aligned} \tag{3.3}$$

and

$$\mathcal{H}_h^d = \max\left(\sum_{a=1}^{A} w_{a,h}\mathcal{F}_a + b_h, 0\right) \tag{3.4}$$

where $w_{a,h}$ is the connection weight from the ath code node to the hth decoder hidden node, $w_{a,i}$ is the connection weight from the ath code node to the ith output node, $w_{h,i}$ is the connection weight from the hth decoder hidden node to the ith output node, and b_i and b_a are biases of the nodes. The width of the output layer equaled the number of items, and the width of the decoder hidden layers was set to 50.

After constructing the structure of the MAEN, the stochastic gradient descent (SGD; Vincent et al., 2008) algorithm was used to train the network using examinees' responses to items by minimizing the cost function shown in Equation (2). When inputting the eth test-taker's response vector $_e$ into the trained MAEN, $\hat{\boldsymbol{\alpha}}_e = \mathcal{F}_e$ can be obtained from the output of the code layer as the estimate of the attribute profile. For example, if $\hat{\boldsymbol{\alpha}}_e = [1, 1, 0]$, the eth examinee is estimated to have mastered attribute 1 and 2 and to have not mastered attribute 3.

3.2.2 Q-matrix Reconstruction Algorithm

The second part of our method is for Q-matrix reconstruction. Since the q-vectors for simple items were assumed to be specified correctly, only the Q-matrix entries for complex items were estimated and reconstructed. Like IRT, CDM also assume local independence (i.e., responses given to the separate items in a test are mutually independent given a certain attribute profile); thus, the q-vectors for all complex items were computed one-by-one. Consider one complex item as an example to describe the reconstruction algorithm. Suppose there were total A attributes measured in the assessment, in other words, the number of columns in the Q-matrix was A. For complex item i, we firstly categorized the students to two groups, $\mathcal{G}_a^0$ and $\mathcal{G}_a^1$, according to the mastery status of ath attribute ($a \in \{1, \ldots, A\}$). Then the proportions of examinees answering the ith item correctly for these two groups, $\bar{X}_i^{\mathcal{G}_a^0}$ and $\bar{X}_i^{\mathcal{G}_a^1}$, are calculated. The difference of $\bar{X}_i^{\mathcal{G}_a^0}$ and $\bar{X}_i^{\mathcal{G}_a^1}$ was denoted as $\Delta p_{i,a}$. After repeating this procedure A times for the ith item, an A-dimensional vector $\boldsymbol{p}_i = \{\Delta p_{i,a}\}$, which was viewed as A observations, could be obtained. $\Delta p_{i,a}$ indicates the difference of correct response rate of ith item between the mastery group and non-mastery group of an attribute a.

If $\Delta p_{i,a}$ was "significantly large", we could infer that the difference of correct response rate of the ith item between the mastery group and non-mastery group was significant and the ith item measures the ath attribute. To determine if $\Delta p_{i,a}$ was "significantly large", we conducted the K-Means clustering (Hartigan & Wong, 1979) on $\boldsymbol{p}_i$. The number of clusters K_i indicated the consistency of the elements in vector $\boldsymbol{p}_i$. For the vector $\boldsymbol{p}_i$, if the intra-vector similarity was very high, the value of K_i is 1, and if the intra-vector similarity was low, K_i is larger than 1. Because $\delta p_{i,a}$ could be either "significantly large" or not "significantly large", K_i could equal either 1 or 2 for each vector $\boldsymbol{p}_i$. If $K_i = 1$, it meant all elements of $\boldsymbol{p}_i$ were "significantly large" or not "significantly large". If $K_i = 2$, some elements of $\boldsymbol{p}_i$ were "significantly large", and the rest were not "significantly large". To determine K_i, we used Calinski-Harabasz pseudo-F statistic (CH-index) as follows:

$$CH(K_i) = \frac{(SS_T - WCSS(K_i))(K_i - 1)}{WCSS(K_i)/(N - Ki)},\ K_i \in \{1, 2\} \tag{3.5}$$

where $WCSS(K_i)$ was the within-cluster sum of square, SS_T was computed as:

$$SS_T = \sum_{a=1}^{A} ||\Delta p_{i,a} - \mu||^2 \tag{3.6}$$

where $\mu = \frac{1}{A}\sum_{a=1}^{A}\Delta p_{i,a}$ was the average of all elements in $\boldsymbol{p}_i$, and A was the number of attributes. The selection of K_i was to maximize the CH index. If $K_i = 1$, all $\Delta p_{i,a}$ were labelled as cluster 1. If $K_i = 2$, we compared the values of the two cluster centers. The cluster center with smaller value was denoted as cluster 1, and the cluster center larger value was denoted as cluster 2. The clustering results of each $\boldsymbol{p}_i$ could be used to reconstruct q-vector $\boldsymbol{q}_i = \{q_{i,a}\}$.

In CDM, $q_i = \{q_{i,a}\}$ can be either 0 or 1, and an item is assumed to measure at least one attribute in an assessment. If $q_{i,a} = 1$, the ith item measures the ath attribute, and the difference of correct response rate between the mastery group and non-mastery group of the ath attribute is significant. While, if $q_{i,a} = 0$, ith item does not measure the ath attribute, and there is no significant difference of correct response rate between the mastery group and non-mastery group of the ath attribute. When $K_i = 1$ in clustering p_i, the ith item measures all attributes (e.g., $q_i = [1, 1, 1]$) because an item measured at least one attribute. When $K_i = 2$, the ith item only measured a subset of attributes (e.g., $q_i = [1, 0, 1]$), and the $q_{i,a}$ was determined as follows:

$$q_{i,a} = \begin{cases} 0, & \text{if } \Delta p_{ia} \text{ belongs to cluster 1} \\ 1, & \text{if } \Delta p_{ia} \text{ belongs to cluster 2} \end{cases} \tag{3.7}$$

For example, for an assessment which measures 3 attributes (i.e., $A = 3$), the q-vector of the ith item may be specified as $q_i = [1, 1, 0]$ after completing the Q-matrix reconstruction. This q-vector indicates that the ith item measures the 1st and 2nd attribute.

3.3 Simulation Study

The aims of the experiment were (1) to examine the attribute profile estimation of the proposed method under different test factors which are expected to affect the estimates' accuracy, (2) to test the performance of Q-matrix reconstruction under these factors, and (3) to compare the proposed method with the performance of two DCMs: the DINA model and the LCDM. Thus, we conducted a simulation study under different assessment conditions with a variety of fixed factors and three manipulated factors.

3.3.1 Design

Fixed factors included the number of attributes measured by the test, the correlation among the attributes, and the sample size. The number of attributes

was 3 with correlations equal to 0.5 between each attribute pair. The number of students was 2000.

Manipulated factors. In the simulation, three factors were manipulated: Test length, test complexity, test quality. Varying these three factors created 24 test conditions in the simulation.

Test length. The first test factor is test length which contained 3 levels. Test lengths of 10, 15 and 20 items are used to represent short, medium and long test conditions, respectively.

Test Complexity. The second test factor is the complexity of the Q-matrix. Simpler Qmatrices include more simple items. We varied the number of simple items per attribute from 1 to 3; this factor could also be viewed as how many identity matrices were in the Q-matrix. The Q-matrix used to generate items for different test conditions is shown in Table 3.1. There are 12 simple items (4 simple items per attribute) and 18 complex items (13, 12, 12 complex items for attribute 1, 2, 3 respectively). Items from this Q-matrix were used to create tests with different test lengths and complexities, based on how many and what types of items were on the test.

Test Diagnostic Quality. The last test factor is test diagnostic quality, which was manipulated by varying test discrimination. We varied test discrimination by varying the proportion of items with high discrimination a test contained, where d_i is calculated as $d_i = p(x = 1|\alpha_1) - p(x = 1|\alpha_0)$. α_0 is the attribute pattern where none of the attributes measured by the ith item are mastered, and α_1 is the attribute pattern where all attributes measured by the ith item are measured. Two levels of item discrimination were examined: high discrimination values were between .3 and .75 ($.3 \leq d_i < .75$); low discrimination values were less than .3 ($0 < d_i < .3$). The reason of choosing .3 as the threshold of the discrimination is based on the previous research studies (Chiu et al., 2009; Cui et al., 2016).

For the simulation, we created two item pools. The items in item pool 1 had high discrimination, and items in item pool 2 were had low discrimination. Both item pools contained 30 items with item-attribute relationships given by the same Q-matrix that is shown in Table 3.1. Tests that varied by diagnostic quality were created by drawing different proportions of items from pool 1 and pool 2, with tests having a higher proportion of items from item pool 1 having better diagnostic quality. Three levels of item discrimination 50%, 70%, and 90% at the test level were set to be low, medium and high levels; these levels represent the proportion of high discrimination items on the test, or test diagnostic quality.

Generating Item Response Probabilities for Item Pools. The true class probabilities of correct response for the items in the item pools were simulated using the logic of a DCM with respect to the Q-matrix defining the item-class relationships and the probabilities following monotonicity constraints across *non-equivalence* classes on an item (i.e., masters of all attributes measured by the item having a higher probability of correct response than masters of a proper subset of these attributes; masters of no attributes measured by the item having a lower probability of correct response than masters of a proper subset of these attributes), but did not follow a particular existing DCM item response function (e.g., the LCDM or DINA function). Current DCM item response functions constrain the item response probabilities to be equal within all equivalence classes; our simulated data did not. Item-based equivalence classes are latent classes that have the same attribute profile, or the same pattern of mastery, for all attributes that are measured by the item. Conversely, item-based non-equivalence classes differ on the mastery status of one or more attributes measured by the item.

We simulated response data using an item by latent class matrix (Xu & Zhang, 2016) as follows:

$$\Pi = \begin{bmatrix} \pi_{1,1} & \pi_{1,2} & \dots & \pi_{1,C} \\ \pi_{2,1} & \pi_{2,2} & \dots & \pi_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{I,1} & \pi_{I,2} & \dots & \pi_{I,C} \end{bmatrix}. \tag{3.8}$$

where the conditional probability that students in cth latent class answer ith item correctly $P(x_i = 1c) = \pi_{i,c}$. I indicated the number of items; C indicated the number of latent classes.

We denote π_{i,α_0} , π_{i,α_1} , and $\pi_{i,\alpha_{\boldsymbol{p}}}$ as the item response probabilities (IRPs) for the nonmastery group, mastery group, and partial mastery group respectively. The mastery group contained students who mastered all of the attributes required by ith item, the partial mastery group contains students who only mastered a proper subset of attributes required by ith item, and the non-mastery group contained students who mastered none of the attributes required by ith item.

As shown in Table 3.2, when simulating response patterns to high discrimination items for item pool 1, for the mastery group π_{i,α_1} were drawn from a uniform distribution $U[.65, .90]$; for the non-mastery group π_{i,α_0} were drawn from a uniform distribution $U[.15, .35]$; and for the partial mastery group $\pi_{i,\alpha_{\boldsymbol{p}}}$ were drawn from a uniform distribution $U[.40, .60]$. These draws yielded an average item discrimination value of .526 for item pool 1 (see values for each item

in Appendix **??**); the drawn values are provided in Appendix **??** for each item and class. When simulating response patterns to low discrimination items for item pool 2, for the non-mastery group π_{i,α_0} were drawn from a uniform distribution $U[.20, .40]$; for partial mastery group $\pi_{i,\alpha_{\boldsymbol{p}}}$ were drawn from a uniform distribution $U[\pi_{i,\alpha_0}, \pi_{i,\alpha_0} + 0.15]$; lastly for the mastery group (students who mastered all the attributes required by ith item) π_{i,α_1} were based on a uniform distribution $U[\pi_{i,\alpha_{\boldsymbol{p}}}, \pi_{i,\alpha_0+.3}]$ for complex items and $U[\pi_{i,\alpha_0}, \pi_{i,\alpha_0}+.3]$ for simple items. This yielded an average item discrimination value of .189 for item pool 2 (see values in Appendix **??**); the drawn values are provided in Appendix **??** for each item and class.

By drawing true item parameters in this way, the $\pi_{i,c}$s in our simulated data differs from IRPs simulated from the LCDM in that partial mastery classes with the same attribute pattern with respect to the measured attributes on a given item (the partial mastery item-based equivalence classes) have different true item response probabilities. The item response probabilities for these classes are, however, drawn from the same uniform distribution, so while they may be different values, they will be in the same range. As an example, consider Item 14 that measures Attribute 2 and 3, as shown in Table 3.1. Classes 3, 4, 5, and 6 are all partial mastery classes with respect to this item: Class 3 and 4 both have mastered Attribute 2 but not Attribute 3, and Class 5 and 6 has both mastered Attribute 3 and not Attribute 2. Under the LCDM, Class 3 and 4 would have the same IRP, while Class 5 and 6 would have the same IRP; under our generating model, the IRP for all four classes were drawn from the same interval, but the draws were different, resulting in, for item pool 1, Class 3 having an IRP of .53, Class 4 having an IRP of .464, Class 5 having an IRP of .462, and Class 6 having an IRP of .444 (see Appendix **??**). For non-mastery equivalence classes and mastery equivalence classes, the true model did constrain draws to be equal within the interval (i.e., Class 1 and 2 have IRP values of .225 and Class 7 and 8 have IRP values of .874). Only for partial mastery item-based equivalence classes were they allowed to differ. The purpose of allowing this difference was to add some noise in the datawhile still controlling the item discrimination level (IRP of mastery group minus IRP of nonmastery group)—so that the LCDM, or any particular DCM, was not perfectly matching the true model.

The values in the item by latent class matrix Π for the two item pools were shown in Appendix **??** and **??**, respectively. The IRPs for the non-mastery group and the mastery group are shown in Appendix B and E. These appendices show this simulation procedure firstly held that $.3 \leq d_i < .75$ for high discrimination item pool (item pool 1) and $0 < d_i < .3$ for low discrimination item pool (item pool 2); it also held for the LCDM primary monotonicity assumptions

(i.e., the mastery group has the greatest IRP, the non-mastery group has the lowest IRP, and the IRP of partial mastery group lay between them).

Generating test forms for different conditions. For each test condition, the items were chosen from the two item pools according to the condition's values for the test discrimination, test complexity, and test length. Item selection ensured the required condition of the proposed method was met. Namely, that for each attribute, at least one simple item with high discrimination measures it. The items that were selected for each test condition are given in Table 3. Items from item pool 1 are presented without an asterisk and items from item pool 2 are presented with an asterisk. For example, when test conditions are medium test length (15 items), have 2 simple items per attribute, and have medium test discrimination (70% of items have high test discrimination), we firstly selected 6 simple items (2 for each attribute) with high discrimination and 4 complex items with high discrimination from item pool 1, then the remaining five complex items with low discrimination were randomly selected from item pool 2. This condition contained 10 items with high discrimination and 5 items with low discrimination (see row 11 in Table 3). In contrast, when test condition are still medium test length (15 items), 2 simple item per attribute, but low test discrimination (50% items are highly discriminative), the assessment consists of 3 highly discriminative items (item 1-3), 3 low discriminating items (item 1*-3*), 4 complex item with high discrimination, and 5 items with low discrimination (row 10 in Table 3.3).

Estimation and analysis procedures. We conducted a comparison between our method and two theoretical DCMs, the LCDM and the DINA model, under the 24 test conditions in the simulation. Each condition was analyzed using 3 models: two DCMs, the LCDM and the DINA model, and the proposed ANN method. For each condition, each DCM was run under two Qmatrix conditions: model fitting with the full, correct Q-matrix and model fitting with the partially-known Q-matrix. When using the partially-known Q-matrix for DCMs model fitting, the response data to the items whose q-vectors were unknown were not used to estimate the parameters in the DCMs. In other words, using the partially-known Q-matrix for DCMs means only using a subset of the items: the simple items with the known q-matrix values. When using the partially known Q-matrix for ANN model fitting, all simple and complex items were used, but only the simple item Q-matrix values were known and used. The Q-matrix entries for the complex items were considered unknown and were missing, or blank. For example, in Table 4, under test condition 1, the number

of items equals to 3 when fitting the DINA model using partially-known Q-matrix (DINA* shown in Table 3.4), but the number of items equals 10 when using the full, correct Q-matrix for DINA (DINA shown in Table 3.4). In that test condition, the ANN method used all 10 items.

Results will be analyzed in terms of classification accuracy of the DCM and ANN model and, for the ANN model, the ability to reconstruct the Q-matrix missing values for the complex items.

The data simulation and DCM model fitting was implemented in R with the "CDM" package (George et al., 2016). The proposed method was programed using the "Tensorflow" library (Abadi et al., 2016) in Python. In the simulation study, we conducted 50 replications. In each replication, the same set of items were used (the selected items under same test conditions as shown in Table 3.3), and new response patterns were created based on the fixed true response probability values given in the item by latent class matrices in Appendices **??** and **??** for each replication.

3.3.2 Experimental Results

First, we tested the effects of the three assessment factors of test length, text complexity, and test diagnostic quality on the accuracy of the attribute profile estimation for proposed method. Then, we compared results from the proposed method to the two completing DCMs, under conditions of complete, accurate Q-matrices and incomplete Q-matrices. Results are given in Tables 3.4, 3.5, and 3.6 for the short, medium, and long test lengths, respectively.

Classification Accuracy and Three Assessment Factors. We first focus on results for the proposed method. Results show that the method works reasonably well and has classification accuracy values greater than 65% for conditions where the test length is 15 or greater, the number of simple items are 2 or greater, and the text complexity has 50% or higher percentage of highly discriminating items. Results show classification accuracy increased in expected ways for the proposed method. Namely, accuracy increases as test length increases, as the number of simple items per attribute increases, and as the test diagnostic quality increases. We can see that for test length, the proposed method achieves classification accuracy greater than 65% under 2 out of 6 test conditions when test length is 10, 6 out of 9 test conditions when test length is 15, and 8 out of 9 test conditions. For simple item per attribute, the proposed method has classification accuracy greater than 65% under 2 out of 9 test conditions when the number of simple item per attribute is 1; under 8 out of 9 test conditions when the number is 2, the proposed method's classification rate is greater than

65%; all classification rates are greater than 65% under 6 test conditions when the number is 3. For the percentage of highly discriminating items, the classification rate is greater than 65% under only 4 out of 8 test conditions when the percentage is 50%; the classification rate is greater than 65% under 6 out of 8 test conditions when the percentage increases to either 70% or 90%.

Next, we examine the results for the DCM methods. Results show that when the Q-matrix is complete and known, the LCDM model works reasonable well and has classification accuracy values greater than 65% for conditions where the test length is 15 or greater, the number of simple items are 2 or greater, and the text complexity has 50% or higher percentage of highly discriminating items (total 17 out of 24 test conditions), and DINO model achieves 65% and greater classification under 13 out of 24 test conditions. When the Q-matrix values are partially known, the accuracy of the methods decrease because only part of items could be used for model fitting. Neither LCDM nor DINA can achieve a 65% classification accuracy when given partially-known Q-matrix. This was negligible (not greater than 5%) for the DINA model under test condition 1, 4, 5, 10, 13, and 22, and for the LCDM model under test condition 4 and 13. This decrease was larger and ranged from 6% to 12% for the DINA model under the rest test conditions and from 6% to 26% for the LCDM model under the rest test conditions.

From the classification results shown in Table 3.4, 3.5, and 3.6, we could observe that longer test length has improved the classification accuracy, even with the additional items were not of highly discriminating level. For example, the number of items with high discrimination contained in the short length test (10 items) with medium test discrimination (70%) equals to the one of the medium length test (15 items) with low test discrimination (50%). Under test condition 2 in Table 4 and test condition 7 in Table 5, the attribute estimation accuracies of the proposed method are [.83, .82, .76, .54] and [.83, .83, .78, .56] corresponding to attribute 1 (A1), attribute 2 (A2), attribute 3 (A3) and attribute pattern (Class). Although either test condition 2 or 7 consisted of 7 highly discriminating items, all accuracies other than attribute 1 (A1) under the second test condition are higher than the first one. The improvement under test condition 7 is caused by the ability of unsupervised learning in exploring the features from the 5 more items whose q-vectors are unknown and discriminating levels are not high.

Comparison Attribute Profiles Estimation with DINA and LCDM. Table 3.4, 3.5, and 3.6 also show the comparison results. LCDM and DINA indicate the results using correct Q-matrix, LCDM* and DINA* indicate results using partial known Q-matrix, and ANN refers to the proposed method.

From this comparison, there are some observations: (1) Like the findings from previous research (Brusco et al., 2017; Cui et al., 2016), our results also prove that when giving the correct Q-matrix, theoretical DCMs always achieve the best estimation accuracy; (2) Only under test condition 4, LCDM using partial Q-matrix can obtain a better classification accuracy (1% higher) than the proposed ANN; in other test conditions, the proposed ANN could achieve more accurate estimation results compared to the theoretical DCMs using partial Q-matrix. This observation shows the power of the proposed unsupervised learning ANN in attribute estimation for large scale assessments without accurate Q-matrix. However, one disadvantage of the proposed ANN method is that the estimation consistence is not as consistent as the theoretical DCMs.

Evaluation of Q-matrix Reconstruction Algorithm. In addition, we tested the effects of the 3 assessment factors on Q-matrix reconstruction (as shown in Table 3.7). The best reconstruction accuracy is 99% (3 simple items per attribute, high diagnostic quality 90%, and medium test length 15). Only 6 of 24 conditions achieved the reconstruction accuracy less than 80%. From the table of reconstruction accuracy under different test conditions (Table 3.7), we can find that the Q-matrix reconstruction results achieved under condition 13 (15, 3, 50%) was better than the one under test condition 22 (20, 3, 50%), because the unknown elements proportion under the first condition is lower than the one under the second condition.

3.4 Conclusion

The object of this paper is to propose an unsupervised learning ANN with fewer constraints according to two potential issues in model based cognitive diagnosis: model selection and Q-matrix misspecification. To achieve this target, we firstly designed a unsupervised learning ANN for attribute estimation which does not rely on specific assumption of item response function and just require partial Q-matrix information (simple items' q-vectors); secondly, we proposed a Q-matrix reconstruction method using K-means clustering algorithms to correct or reconstruct the mis-specified or missing elements of the Q-matrix.

We tested our methodology and compared with two theoretical DCMs (DINA and LCDM) under 24 types of simulated test conditions according to test length, number of simple items per attribute, and the test discriminating. The results showed some advantages of our proposed method.

The primary advantage of the proposed method is that it provides an option for users to apply cognitive diagnostic classification to large scale assessment

responses data when lacking prior knowledge about a part of the items (i.e., unknown Q-matrix). The proposed method does not rely on pre data analysis to determine the Q-matrix before doing diagnostic classification and provide a reasonable classification accuracy.

In addition to being able to get models results without full item information, the second advantage of the proposed method is that it has the ability to reconstruct the item-attribute relationship (Q-matrix) only using limited information. The results show that the method could reconstruct well when the test contains 15 or more items and 70% or higher diagnostic quality.

Both the first and second advantages make the outputs (e.g., classification and reconstructed Q-matrix) of the proposed method be able to help to determine which DCM is the appropriate model for the current data. In other words, when we have the classification results from ANN results, for each item we could choose the appropriate item response function and determine which DCM is the true model by considering the reconstructed Q-matrix and comparing the item response probability of different groups.

The last advantage of the proposed methodology demonstrated by the experimental results is that unlike the typical DCMs which removes the items with low discriminating power or the items with unknown q-vector, the proposed method don't removing such items because the MAEN structure in the proposed method has the ability to explore useful information from these items which cannot be used in typical DCMs.

3.5 Discussion

The study demonstrates promise for using unsupervised learning artificial neural networks to achieve cognitive diagnostic classification and Q-matrix reconstruction. However, our method has some limitations. First, this method cannot determine the number of attributes. We assumed the number of attributes was known and accurate in this study. Second, the proposed method requires at least 1 simple item for each attribute. Thus, the proposed method cannot be used as a method to determine the number of attributes, or to explore the item attribute relationship if the three assumptions (i.e., number of attributes is known, Q-matrix of all simple items is known, at least a high discriminating simple item measures each attribute) cannot be hold. Third, as shown in the experimental results, the classification accuracy is not as high as the DINA and LCDM when the Q-matrix is completely known. Thus, when item-attribute information is known, the proposed method does not have an advantage over existing DCMs.

In addition to the attribute profile estimation method, there is one concern for the Q-matrix reconstruction method. Although the experimental results showed the propose K-means based method achieved a reasonable reconstruction accuracy for Q-matrix unknown elements, but the K-means assumptions (e.g., each cluster has roughly equal number of observations, the variance of the distribution of each variable is spherical) are hard to be hold because the input size of the K-means, which equals to the number of attributes, is very small (i.e., 3 in this study). It means that the proposed Q-matrix reconstruction cannot be always guaranteed to be applied successfully.

In this study, we consider a small number of key factors to manipulate and examine. Other factors may also be examined. For example, we did not consider a hierarchical attribute structure. The proposed method may or may not be applicable when such structure is present and future research is needed to understand its performance in that scenario. Other future research may focus on the influence of other factors of diagnostic assessment design on the method's performance.

In addition, future research may extend this topic in three directions. The first direction would be to extend the methodology to have the capability of determining the number of attributes. This would remove the assumption that the number of attributes is known. This topic will be very helpful when doing explorative analysis on a new assessment. One potential method is to introduce the number of attributes as an hyperparameters into constructing artificial neural networks and then using validating test to determine the appropriate value.

A second direction would be to adjust the activation function. In the proposed method, we used the sigmoid function (logistic function) as the activation function of the output nodes. Although for complex items, we added an hidden layer to approximate the relationship between attribute profile and the item response probability, for simple items, the reconstruction of the item response is still using a logit probability (i.e., we choose the sigmoid function as the activation function of the output nodes in the ANNs). Such item response function may not be accurate because it still holds a constrain that the IRP of classes with the same attribute pattern with respect to the measured attributes on a given simple item are equal. In the future, the activation function could be replaced using a softmax function to convert the binary classification to a multiple classification and would be expected to increase the classification accuracy because the softmax activation function could provide each class a unique IRP with regard to an item.

In addition, considering the concern of the proposed K-means based Q-matrix reconstruction method, in the future, the Q-matrix reconstruction method

would be updated by directly using the trained ANN. One potential further application is to tune the values of the code layer nodes and compare the output results. Taking the test measures 3 attributes as an example, we firstly give an attribute profiles [0, 0, 0] as the input of output layer and get the IRP for all items from the output of the trained ANN. Then, we give an attribute profiles [0, 0, 1] as the input of output layer and get the second IRP for all items. By comparing these two IRPs, we could determine the relationship between the Attributes 3 and items and figure out which item measures the third attribute.

Lastly, the method may be extended to have the capability to detect hierarchical attribute structures. To achieve this, the Q-matrix reconstruction component of the method would be redesigned to determine the attributes relationships by using classifications from the artificial neural networks. Then, such attribute relationships can be used to adjust the connection between the output nodes and attribute nodes, which would better model the assessment data and, thus, lead to improved classification accuracy for test-takers.

Table 3.1: Q-matrix for creating two item pools.

Item ID	Attribute 1	Attribute 2	Attribute 3
Item 1	1	0	0
Item 2	0	1	0
Item 3	0	0	1
Item 4	1	0	0
Item 5	0	1	0
Item 6	0	0	1
Item 7	1	0	1
Item 8	0	1	1
Item 9	1	1	0
Item 10	0	1	1
Item 11	1	1	0
Item 12	1	0	1
Item 13	1	1	0
Item 14	0	1	1
Item 15	1	0	1
Item 16	1	1	0
Item 17	0	1	1
Item 18	1	0	1
Item 19	1	1	1
Item 20	1	0	0
Item 21	0	1	0
Item 22	0	0	1
Item 23	0	1	1
Item 24	1	0	1
Item 25	1	1	0
Item 26	1	0	1
Item 27	0	0	1
Item 28	0	1	0
Item 29	1	1	0
Item 30	1	0	0

Table 3.2: The table of selecting $\pi_{i,c}$ for item by class matrix.

Latent Groups	High Discrimination	Low Discrimination
Non-mastery π_{i,α_0}	$\mathbf{U}[.15, .35]$	$\mathbf{U}[.20, .40]$
Partial-mastery $\pi_{i,\alpha_{\boldsymbol{p}}}$	$\mathbf{U}[.40, .60]$	$\mathbf{U}[\pi_{i,\alpha_0}, \pi_{i,\alpha_0} + .15]$
Mastery π_{i,α_1}	$\mathbf{U}[.65, .90]$	$\begin{cases} U[\pi_{i,\alpha_{\boldsymbol{p}}}, \pi_{i,\alpha_0}+.30], & \text{complex items} \\ U[\pi_{i,\alpha_0}, \pi_{i,\alpha_0} + .30], & \text{simple items} \end{cases}$

Note. For each item, π_{i,α_0}, π_{i,α_0}, and $\pi_{i,\alpha_{\boldsymbol{p}}}$ indicate the $\pi_{i,c}$ for non-mastery group, mastery group, and partial mastery group, respectively.

Table 3.3: Item selection under different test conditions.

Test Length	# Simple Items Per Attribute	Discrimination (% High Discrim)	Item ID	
			Simple Item	Complex Item
10	1	50%	1:3	7, 8, 9*:13*
		70%	1:3	7:10, 11*:13*
		90%	1:3	7:12, 13*
	2	50%	1:3, 1*:3*	7, 8*:10*
		70%	1:6	7, 8*:10*
		90%	1:6	7:9, 10*
15	1	50%	1:3	7:10, 7*:14*
		70%	1:3	7:13, 7*:11*
		90%	1:3	7:16, 7*, 8*
	2	50%	1:3, 4*:6*	7:10, 7*:11*
		70%	1:6	7:10, 7*:11*
		90%	1:6	7:13, 7*, 8*
	3	50%	1:6, 1*:3*	7, 7*:11*
		70%	1:6, 20:22	7, 7*:11*
		90%	1:6, 20:22	7:10, 7*, 8*
20	1	50%	1:3	7:13, 7*:16*
		70%	1:3	7:17, 7*:12*
		90%	1:3	7:19, 23, 24, 7*, 8*
	2	50%	1:6	7:10, 7*:16*
		70%	1:6	7:14, 7*:12*
		90%	1:6	7:18, 7*, 8*
	3	50%	1:6, 20:22	7, 7*:16*
		70%	1:6, 20:22	7:11, 7*:12*
		90%	1:6, 20:22	7:15, 7*, 8*

Note. * indicates the item ID of items with low discrimination (item pool 2).

Table 3.4: Comparisons of attribute estimation accuracy under the short assessment length (10 items).

Test Condition	# Simple Itmes per Attribute	Discrim (% High Discrim)	Methods	# Items for Model Fitting	Attribute 1	Attribute 2	Attribute 3	Class
1	1	50%	ANN	10	.82 (.04)	.81 (.03)	.75 (.03)	.50 (.03)
			DINA	10	.81 (.01)	.81 (.01)	.75 (.00)	.50 (.00)
			DINA*	3	.82 (.00)	.82 (.00)	.72 (.01)	.47 (.01)
			LCDM	10	.84 (.01)	.84 (.00)	.78 (.00)	.54 (.00)
			LCDM*	3	.77 (.01)	.78 (.01)	.69 (.00)	.45 (.01)
2		70%	ANN	10	.83 (.03)	.82 (.04)	.76 (.03)	.54 (.04)
			DINA	10	.85 (.01)	.84 (.00)	.75 (.01)	.54 (.01)
			DINA*	3	.82 (.00)	.82 (.00)	.72 (.01)	.47 (.01)
			LCDM	10	.85 (.01)	.85 (.00)	.80 (.00)	.57 (.00)
			LCDM*	3	.77 (.01)	.78 (.01)	.69 (.00)	.45 (.01)
3		90%	ANN	10	.85 (.03)	.85 (.03)	.79 (.03)	.57 (.03)
			DINA	10	.84 (.01)	.83 (.00)	.76 (.00)	.54 (.00)
			DINA*	3	.82 (.00)	.82 (.00)	.72 (.01)	.47 (.01)
			LCDM	10	.86 (.01)	.85 (.00)	.81 (.00)	.61 (.01)
			LCDM*	3	.77 (.01)	.78 (.01)	.69 (.00)	.45 (.01)
4	2	50%	ANN	10	.88 (.03)	.81 (.02)	.81 (.03)	.60 (.03)
			DINA	10	.89 (.02)	.82 (.00)	.82 (.01)	.63 (.01)
			DINA*	6	.88 (.01)	.82 (.01)	.81 (.00)	.60 (.01)
			LCDM	10	.90 (.00)	.83 (.00)	.84 (.00)	.64 (.00)
			LCDM*	6	.88 (.01)	.82 (.01)	.81 (.00)	.61 (.01)
5		70%	ANN	10	.90 (.03)	.84 (.02)	.83 (.02)	.65 (.02)
			DINA	10	.90 (.01)	.84 (.00)	.84 (.00)	.65 (.00)
			DINA*	6	.88 (.01)	.82 (.01)	.81 (.00)	.60 (.01)
			LCDM	10	.90 (.00)	.84 (.00)	.85 (.01)	.67 (.00)
			LCDM*	6	.88 (.01)	.82 (.01)	.81 (.00)	.61 (.01)
6		90%	ANN	10	.91 (.02)	.88 (.02)	.86 (.02)	.70 (.03)
			DINA	10	.90 (.01)	.87 (.00)	.85 (.00)	.68 (.01)
			DINA*	6	.88 (.01)	.82 (.01)	.81 (.00)	.60 (.01)
			LCDM	10	.92 (.00)	.89 (.00)	.88 (.01)	.73 (.00)
			LCDM*	6	.88 (.01)	.82 (.01)	.81 (.00)	.61 (.01)

Note. "# Items for Model Fitting" indicates the number of items used to fit the model for each method. ANN: the proposed MAEN; DINA: DINA fitting with correct Q-matrix; LCDM: LCDM fitting with correct Q-matrix; DINA*: DINA model fitting with partially-known Q-matrix; LCDM*: LCDM model fitting with partially-known Q-matrix.

Table 3.5: Comparisons of attribute estimation accuracy under the short assessment length (15 items).

Test Condition	# Simple Itmes per Attribute	Discrim (% High Discrim)	Methods	# Items for Model Fitting	Attribute 1	Attribute 2	Attribute 3	Class
7	1	50%	ANN	15	.83 (.03)	.83 (.03)	.78 (.03)	.56 (.03)
			DINA	15	.83 (.01)	.82 (.00)	.78 (.01)	.56 (.01)
			DINA*	3	.82 (.01)	.82 (.00)	.72 (.00)	.47 (.01)
			LCDM	15	.85 (.00)	.84 (.00)	.80 (.00)	.59 (.00)
			LCDM*	3	.77 (.01)	.78 (.00)	.69 (.00)	.45 (.01)
8		70%	ANN	15	.85 (.03)	.85 (.02)	.81 (.03)	.60 (.04)
			DINA	15	.84 (.01)	.85 (.00)	.79 (.00)	.58 (.00)
			DINA*	3	.82 (.01)	.82 (.00)	.72 (.00)	.47 (.01)
			LCDM	15	.88 (.00)	.86 (.00)	.84 (.00)	.63 (.00)
			LCDM*	3	.77 (.01)	.78 (.00)	.69 (.00)	.45 (.01)
9		90%	ANN	15	.86 (.02)	.87 (.03)	.83 (.03)	.62 (.03)
			DINA	15	.84 (.00)	.85 (.01)	.78 (.01)	.57 (.00)
			DINA*	3	.82 (.01)	.82 (.00)	.72 (.00)	.47 (.01)
			LCDM	15	.88 (.00)	.88 (.00)	.85 (.00)	.66 (.00)
			LCDM*	3	.77 (.01)	.78 (.00)	.69 (.00)	.45 (.01)
10	2	50%	ANN	15	.89 (.03)	.84 (.02)	.83 (.02)	.65 (.03)
			DINA	15	.89 (.00)	.83 (.00)	.82 (.01)	.63 (.00)
			DINA*	6	.88 (.02)	.82 (.02)	.81 (.00)	.62 (.02)
			LCDM	15	.91 (.00)	.85 (.00)	.85 (.01)	.68 (.00)
			LCDM*	6	.88 (.01)	.82 (.01)	.81 (.00)	.62 (.01)
11		70%	ANN	15	.91 (.02)	.86 (.02)	.86 (.01)	.68 (.03)
			DINA	15	.91 (.00)	.86 (.00)	.85 (.01)	.66 (.01)
			DINA*	6	.88 (.02)	.82 (.02)	.81 (.00)	.62 (.02)
			LCDM	15	.92 (.00)	.87 (.00)	.88 (.01)	.69 (.00)
			LCDM*	6	.88 (.01)	.82 (.01)	.81 (.00)	.62 (.01)
12		90%	ANN	15	.92 (.01)	.88 (.03)	.89 (.02)	.71 (.03)
			DINA	15	.91 (.00)	.87 (.00)	.87 (.00)	.68 (.01)
			DINA*	6	.88 (.02)	.82 (.02)	.81 (.00)	.62 (.02)
			LCDM	15	.92 (.00)	.91 (.00)	.91 (.01)	.73 (.00)
			LCDM*	6	.88 (.01)	.82 (.01)	.81 (.00)	.62 (.01)
13	3	50%	ANN	15	.90 (.02)	.85 (.03)	.86 (.02)	.66 (.02)
			DINA	15	.90 (.00)	.86 (.01)	.86 (.00)	.67 (.00)
			DINA*	9	.89 (.00)	.84 (.01)	.82 (.01)	.64 (.01)
			LCDM	15	.90 (.00)	.86 (.01)	.86 (.00)	.67 (.00)
			LCDM*	9	.89 (.02)	.84 (.01)	.82 (.01)	.63 (.01)
14		70%	ANN	15	.91 (.01)	.88 (.02)	.87 (.02)	.70 (.02)
			DINA	15	.91 (.00)	.88 (.00)	.88 (.01)	.71 (.00)
			DINA*	9	.89 (.00)	.84 (.01)	.82 (.01)	.64 (.01)
			LCDM	15	.92 (.00)	.88 (.00)	.89 (.00)	.73 (.00)
			LCDM*	9	.89 (.02)	.84 (.01)	.82 (.01)	.63 (.01)
15		90%	ANN	15	.92 (.01)	.90 (.01)	.89 (.01)	.73 (.01)
			DINA	15	.91 (.01)	.88 (.00)	.88 (.00)	.71 (.00)
			DINA*	9	.89 (.00)	.84 (.01)	.82 (.01)	.64 (.01)
			LCDM	15	.93 (.00)	.91 (.01)	.90 (.00)	.74 (.00)
			LCDM*	9	.89 (.02)	.84 (.01)	.82 (.01)	.63 (.01)

Table 3.6: Comparisons of attribute estimation accuracy under the short assessment length (20 items).

Test Condition	# Simple Itmes per Attribute	Discrim (% High Discrim)	Methods	# Items for Model Fitting	Attribute 1	Attribute 2	Attribute 3	Class
16	1	50%	ANN	20	.86 (.03)	.86 (.02)	.82 (.03)	.58 (.04)
			DINA	20	.87 (.00)	.84 (.01)	.80 (.01)	.57 (.02)
			DINA*	3	.82 (.02)	.82 (.01)	.72 (.01)	.47 (.01)
			LCDM	20	.89 (.00)	.87 (.01)	.84 (.01)	.63 (.01)
			LCDM*	3	.77 (.01)	.78 (.01)	.69 (.00)	.45 (.01)
17		70%	ANN	20	.87 (.02)	.88 (.03)	.84 (.02)	.66 (.03)
			DINA	20	.84 (.00)	.87 (.00)	.80 (.01)	.59 (.00)
			DINA*	3	.82 (.02)	.82 (.01)	.72 (.01)	.47 (.01)
			LCDM	20	.89 (.00)	.89 (.00)	.87 (.00)	.69 (.00)
			LCDM*	3	.77 (.01)	.78 (.01)	.69 (.00)	.45 (.01)
18		90%	ANN	20	.87 (.03)	.87 (.02)	.86 (.03)	.67 (.03)
			DINA	20	.84 (.01)	.86 (.00)	.80 (.00)	.59 (.01)
			DINA*	3	.82 (.02)	.82 (.01)	.72 (.01)	.47 (.01)
			LCDM	20	.89 (.01)	.89 (.00)	.89 (.01)	.71 (.01)
			LCDM*	3	.77 (.01)	.78 (.01)	.69 (.00)	.45 (.01)
19	2	50%	ANN	20	.91 (.04)	.86 (.04)	.86 (.02)	.69 (.03)
			DINA	20	.91 (.01)	.85 (.01)	.85 (.01)	.65 (.01)
			DINA*	6	.88 (.01)	.82 (.00)	.81 (.02)	.62 (.01)
			LCDM	20	.91 (.00)	.86 (.01)	.88 (.00)	.70 (.00)
			LCDM*	6	.88 (.02)	.82 (.02)	.81 (.01)	.62 (.01)
20		70%	ANN	20	.92 (.03)	.88 (.02)	.88 (.03)	.71 (.03)
			DINA	20	.91 (.00)	.88 (.01)	.86 (.00)	.69 (.01)
			DINA*	6	.88 (.01)	.82 (.00)	.81 (.02)	.62 (.01)
			LCDM	20	.93 (.00)	.89 (.00)	.88 (.00)	.73 (.00)
			LCDM*	6	.88 (.02)	.82 (.02)	.81 (.01)	.62 (.01)
21		90%	ANN	20	.92 (.03)	.91 (.02)	.90 (.02)	.74 (.03)
			DINA	20	.92 (.01)	.88 (.00)	.87 (.00)	.70 (.01)
			DINA*	3	.88 (.01)	.82 (.00)	.81 (.02)	.62 (.01)
			LCDM	20	.93 (.01)	.92 (.00)	.91 (.00)	.77 (.01)
			LCDM*	3	.88 (.02)	.82 (.02)	.81 (.01)	.62 (.01)
22	3	50%	ANN	20	.92 (.02)	.88 (.02)	.88 (.03)	.69 (.03)
			DINA	20	.91 (.01)	.86 (.01)	.87 (.00)	.68 (.01)
			DINA*	9	.89 (.02)	.84 (.01)	.82 (.02)	.64 (.02)
			LCDM	20	.92 (.00)	.87 (.01)	.88 (.00)	.72 (.00)
			LCDM*	9	.89 (.02)	.84 (.02)	.82 (.01)	.63 (.01)
23		70%	ANN	20	.92 (.03)	.90 (.02)	.88 (.02)	.73 (.03)
			DINA	20	.92 (.01)	.89 (.00)	.87 (.01)	.71 (.01)
			DINA*	9	.89 (.02)	.84 (.01)	.82 (.02)	.64 (.02)
			LCDM	20	.93 (.00)	.89 (.01)	.89 (.00)	.74 (.01)
			LCDM*	9	.89 (.02)	.84 (.02)	.82 (.01)	.63 (.01)
20		90%	ANN	20	.92 (.03)	.91 (.02)	.90 (.03)	.77 (.03)
			DINA	20	.91 (.01)	.88 (.00)	.88 (.00)	.72 (.01)
			DINA*	9	.89 (.02)	.84 (.01)	.82 (.02)	.64 (.02)
			LCDM	20	.93 (.00)	.92 (.00)	.92 (.00)	.79 (.00)
			LCDM*	9	.89 (.02)	.84 (.02)	.82 (.01)	.63 (.01)

Table 3.7: Q-matrix reconstruction accuracy under different assessment conditions.

Test Condition	Test Length	# Simple Items Per Attribute	Discrim (% High Discrim)	Column 1 (Attribute 1)	Column 2 (Attribute 2)	Column 3 (Attribute 3)	Q-matrix
1	10	1	50%	.58 (.05)	.70 (.04)	.58 (.02)	.62 (.03)
2			70%	.80 (.02)	.75 (.03)	.66 (.02)	.72 (.03)
3			90%	.86 (.01)	.85 (.01)	.82 (.02)	.84 (.02)
4		2	50%	.81 (.02)	.67 (.02)	.60 (.03)	.68 (.03)
5			70%	.89 (.01)	.82 (.01)	.83 (.03)	.81 (.02)
6			90%	.99 (.01)	.97 (.01)	.99 (.00)	.98 (.01)
7	15	1	50%	.72 (.02)	.72 (.03)	.68 (.01)	.71 (.01)
8			70%	.80 (.00)	.76 (.02)	.77 (.01)	.78 (.02)
9			90%	.85 (.01)	.88 (.01)	.86 (.00)	.87 (.01)
10		2	50%	.84 (.01)	.83 (.03)	.87 (.04)	.85 (.02)
11			70%	.92 (.04)	.86 (.03)	.87 (.03)	.87 (.05)
12			90%	.94 (.03)	.93 (.00)	.98 (.00)	.95 (.02)
13		3	50%	.95 (.02)	.93 (.02)	.97 (.01)	.95 (.01)
14			70%	.96 (.01)	.96 (.01)	.97 (.01)	.97 (.01)
15			90%	.99 (.00)	.99 (.00)	.99 (.00)	.99 (.01)
16	20	1	50%	.75 (.03)	.75 (.03)	.74 (.04)	.75 (.03)
17			70%	.81 (.04)	.82 (.01)	.80 (.02)	.81 (.01)
18			90%	.86 (.02)	.86 (.01)	.87 (.02)	.86 (.01)
19		2	50%	.78 (.03)	.74 (.02)	.75 (.04)	.80 (.00)
20			70%	.93 (.01)	.93 (.01)	.90 (.02)	.93 (.01)
21			90%	.94 (.01)	.90 (.01)	.97 (.00)	.94 (.01)
22		3	50%	.83 (.01)	.80 (.03)	.83 (.03)	.85 (.03)
23			70%	.97 (.01)	.96 (.01)	.98 (.00)	.97 (.02)
24			90%	.97 (.00)	.97 (.01)	.99 (.00)	.98 (.01)

Chapter 4

A Semi-Supervised Learning-based Diagnostic Classification Method using Artificial Neural Networks

4.1 Introduction

The purpose of cognitive diagnostic modelling (CDM; J. L. Templin and Henson, 2006) or diagnostic measurement is to provide students' skill/knowledge/attributes mastery status (mastery or non-mastery) through their responses to items from carefully designed assessments. Because of the ability to provide educators diagnostic feedback from students' assessment results, CDM has been the focus of much research in the last decade. Various types of theoretical diagnostic classification models (TDCMs), such as the deterministic inputs, noisy and gate (DINA; Junker and Sijtsma, 2001), the reparametrized unified model/fusion model (RUM; Hartz, 2002) and the log-linear cognitive diagnosis model (LCDM; Henson et al., 2009), are designed based on different cognitive theories or assumptions about the relationship between a student's response pattern and attribute profile.

When analysing a particular assessment dataset, selecting inappropriate TDCMs (model misspecification) impacts the classification accuracy and parameter estimation. For example, when the attributes measured by an assessment

are non-compensatory, which indicates that nonmastery on one attribute cannot be compensated by mastery on another attribute, selecting a compensatory model will decrease the performance of classification and measurement. DINA model and DINO (J. L. Templin & Henson, 2006) model achieved worse fit than did the other more relaxed TDCMs, such as G-DINA (DeCarlo, 2011), LCDM, and RUM because both DINA and DINO might be too restrictive to reflect actual students' knowledge status (Yamaguchi & Okada, 2018). Thus, a principal research question of the previous research studies in CDM is which model better describes the data.

A Q-matrix indicates the relationship between items and attributes in an assessment. Q-matrices are often carefully designed by assessment experts, whereas some existing research and their experimental results have shown that Q-matrices constructed by content experts do not always reflect the relationship precisely and may require empirically-driven modifications (Bradshaw et al., 2014; Tjoe & de la Torre, 2014). In CDM, the diagnostic quality of an item indicates the discriminating power of the item to determine the success of the diagnosis. The item with high discriminating refers to that students who have mastered the attributes required by the item are expected to have a high probability of responding to the item correctly, while students who have not are expected to have a low probability. Items with low discriminating power compromise the accuracy of the estimate of student attribute mastery. In the previous research studies, the performances of all TDCMs are sensitive to either the diagnostic quality of items or the accuracy of Q-matrices (Kunina-Habenicht et al., 2012; R. Liu et al., 2017).

Because of the increase of data size and development of computational power, artificial neural networks (ANNs; I. Goodfellow et al., 2016) have been proposed as an attractive approach to convert a pattern of item responses into a diagnostic classification (Cui et al., 2017; Cui et al., 2016; Paulsen, 2019; Xue, 2019). An ANN is a computational system inspired by biological neural systems for information processing in animals' brains. An ANN is built on inputs being translated to outputs through a series of neuron layers. It consists of three types of layers: an input layer, hidden layer(s), and an output layer. Each layer consists of a number of neurons (or nodes), and each node is connected to the nodes in the next layer. Each layer (except for the input layer) uses the output of its previous layer as the input. Supervised learning ANNs were applied in some research studies (Cui et al., 2017; Cui et al., 2016; Paulsen, 2019). To train the supervised learning ANNs, the ideal response patterns were set as the input layer and the associated attribute profiles as the output layer. Cui et al. (2016) hypothesized DINA model with both slipping and guessing equalling to 0 to synthesize ideal

responses to train a multilayer perceptron (MLP). The experimental results showed that the classification accuracy of the supervised learning ANNs was not appreciated even in the simulated study. Another disadvantage of applying supervised learning ANNs for CDM is how to create the ideal response patterns using a TDCM because both TDCM and parameters are difficult to hypothesize. In addition to supervised learning ANNs, Cui et al. (2016) used one type of unsupervised learning ANNs, self-organizing map (SOM), to classify test-takers into different latent groups for CDM. One disadvantage of the unsupervised learning ANNs is that some further data analysis approaches are required to label the clusters. For example, although cluster analysis can place test-takers into different latent groups, post hoc techniques are required to discern the attributes from these latent groups. To do cluster labelling, Xue (2018) proposed a modified autoencoder network with a sparsely connected decoder explained the code layer outputs by using a part of the Q-matrix information. However, in both research studies, the unsupervised learning ANNs cannot yield comparable classification results compared with the TDCMs, especially when the diagnostic quality of the assessment was not high. In addition, the ANNs methods produced very unstable and unappreciated estimation unless a great deal of care was taken to conduct sensitivity analyses (Briggs & Circi, 2017).

Regarding the disadvantages in supervised leaning ANNs and unsupervised learning ANNs, in this research, the purpose of using semi-supervised learning thinking is to provide reasonable labels for ANN training and provide accurate and robust classification under different test conditions. In this research, we firstly applied the semi-supervised learning thinking into the ANNs-based CDM. Two typical TDCMs, DINA model and DINO model, were contained in the proposed framework to improve the accuracy and consistency of the ANN's classification. In the following sections, we will firstly give a brief introduction to semi-supervised learning and the Co-Training method we used in this framework. Then, the structure of the ANN will be described. Additionally, the experimental results under both simulated experiments are illustrated to compare the proposed method and 5 different TDCMs. Lastly, the benefits and challenges of this methodology are summarized, and future research is also outlined.

4.1.1 Semi-supervised Learning

In the machine learning field, semi-supervised learning (X. J. Zhu, 2005) concerns the study of how computers and natural systems learn in the presence of both labelled and unlabelled data, and it is somewhere between supervised learn-

ing and unsupervised learning. The research goal of semi-supervised learning is to understand how combining labelled and unlabelled data change the learning behaviour, and design algorithms that take advantage of such a combination. Semisupervised learning is a great interest in a wide range of applications, such as image search (Fergus et al., 2009), natural language parsing (Liang, 2005), and speech analysis (Y. Liu & Kirchhoff, 2014) because the labelled data is scarce or expensive.

In semi-supervised learning, to handle the incomplete labels, the most straightforward algorithm for semi-supervised learning is based on a self-training (Haffari & Sarkar, 2012; Rosenberg et al., 2005) scheme using bootstrapping with additional labelled data obtained from its own highly confident prediction. Graph-based semi-supervised learning (Fergus et al., 2009) models are applied through efficient spectral methods requiring eigen analysis of the graph Laplacian and have been shown to be one of the most effective approaches for classification tasks from a wide range of domains. Transudative SVMs (Joachims, 1999) extend SVMs with the aim of max-margin classification while ensuring that there are as few unlabelled observations near the margin as possible. More recently, the techniques to solve the training using noisy labels using artificial neural networks have begun to receive attention, such as Restricted Boltzmann Machine (RBM; Larochelle and Bengio, 2008) and Generative Stochastic Networks (Bengio et al., 2014); Mnih and Hinton (2012) also developed the deep neural network with robust loss function to handle label-omission and registration error.

4.2 Method

4.2.1 Co-Training Methods of using DINA Model and DINO Model

As one typical semi-supervised learning method, Co-Training (Nigam & Ghani, 2000) methods use a pair of classifiers with separate views of the data to iteratively learn and generate additional training labels. Like the self-training scheme, Co-Training is a wrapper method and widely applicable to many tasks. Co-Training bears a strong resemblance to the self-training scheme because each classifier uses its most confident predictions on unlabelled instances to teach itself. Two classifiers operate on different views of one observation, and the success of CoTraining depends on the following two assumptions (X. Zhu & Goldberg, 2009): 1) each view alone is sufficient to make good classifications,

given enough labelled data; 2) the two views are conditionally independent given the class label.

Inspired by the typical Co-Training method, in this research, we chose the DINA model and DINO model as two classifiers to operate on different views of one response pattern to an item. The DINA model is a non-compensatory, or conjunctive TDCM means that a lack of one attribute cannot be compensated by the mastery of another attribute measured by an item. For each item, the DINA model classifies students into two groups: those who have mastered all the attributes required by the item and those who have not. The jth item response probability of the ith student can be written as:

$$P(y_{ij} = 1|\xi ij, s_j, g_j) = (1 - s_j)^{\xi ij} g_j^{1-\xi ij} \tag{4.1}$$

where $\xi_{ij} = 1$ indicates the ith student has mastered all required attributes of jth item, and $\xi_{ij} = 0$ refers to non-mastery status; s_j and g_j are the slipping parameter and guessing parameter of the jth item.

In contrast to the DINA model, the DINO model is a compensatory or disjunctive TDCM, which means that a non-mastery on one latent attribute can be compensated for by a mastery status on another attribute. The jth item response probability of the ith student can be written as:

$$P(y_{ij} = 1|\omega ij, s_j, g_j) = (1 - s_j)^{\omega ij} g_j^{1-\omega ij} \tag{4.2}$$

where the latent response $\omega_{ij} = 0$ indicates that the ith student has mastered at least one attribute measured by jth item, and $\omega_{ij} = 1$ indicates the absence of all required attributes. Like DINA, s_j and g_j are the slipping parameter and guessing parameter of the jth item.

Regarding the two assumptions of successfully applying Co-Training method, the reasons for selecting the DINA model and the DINO model are two-fold. First, in an assessment, either the DINA model or the DINO model can be a true model for different items. For example, both the DINA and DINO models are true models for a simple structure item, which only measures a single attribute. Thus, using either the DINA model or the DINO model is sufficient to make reasonable classification results. The second reason is that as described above, the item response functions of the DINA model and DINO model are represented based on different assumptions on the relationship between response patterns and attribute profiles. In other words, the classification results of using the DINA model and the DINO model are independent.

In this paper, given the response data and Q-matrix, the DINA model and the DINO model were fitted. For an individual test-taker, we use two labels

c_{DINA} and c_{DINO}. c_{DINA} was the estimated latent class under the assumption of using the DINA model, and c_{DINO} was the estimated latent class under the assumption of using the DINO model. c_{DINA} and c_{DINO} could be either the same or different. In this research, the One-Hot encoding method was applied to the integer encoding c_{DINA} and c_{DINO} to create two new One-Hot representation vectors $\boldsymbol{c}_{DINA} = \{c^k_{DINA}\}$ and $\boldsymbol{c}_{DINA} = \{c^{k'}_{DINA}\}$. c^k_{DINA} and $c^{k'}_{DINA} \in \{0, 1\}$, and $\sum_k c^k_{DINA} = \sum_{k'} c^{k'}_{DINA} = 1$. For example, if there are 4 latent classes, the integer encoding labels 1, 2, 3 and 4 are converted to One-Hot encoding [0001], [0010], [0100], and [1000], respectively.

4.2.2 Semi-supervised Learning ANN for Diagnostic Classification

As shown in Figure 4.1, like the supervised learning ANNs, the proposed semi-supervised learning ANN also consisted of three parts: the input layer, hidden layers, and the output layer. The number of nodes (the circles in Figure 4.1) on the input layer was equal to the number of items contained in the assessment. The number of nodes on the output layer was equal to the number of latent classes. To establish the relationship between the input and output nodes, we used three hidden layers to convert observed response patterns to latent classes. The numbers of nodes at these two hidden layers are 200, 100, and the number of latent classes, respectively. We use the Rectified linear unit (ReLU; I. Goodfellow et al., 2016) as the activation function for the first two hidden layers and softmax function as the activation function for the third hidden layer. In the supervised learning ANNs, the number of output nodes was equal to the number of latent classes or number of attributes. The output layers in our proposed semi-supervised learning ANN consisted of two parts. The first part (output 1) corresponded to the DINA model classification, and the second part (output 2) corresponded to the DINO model classification. The total number of output nodes was equal to two times of the number of hidden classes. For example, consider an assessment with 30 items that measured a total of 4 attributes, the input layer consisted of 30 input nodes (30 items), the third hidden layer consisted of 16 nodes ($2\times4 = 16$ latent classes), and the output layer consisted of 32 nodes (2×16).

In the supervised learning ANNs in CDM, only a single label was used for each observation. For example, when only using DINA classification as labels, the supervised learning ANN was used to train the standard softmax regression or a sigmoid regression (Pang et al., 2020) inputs to outputs without taking into account incorrect labels. In contrast, the proposed semi-supervised learning

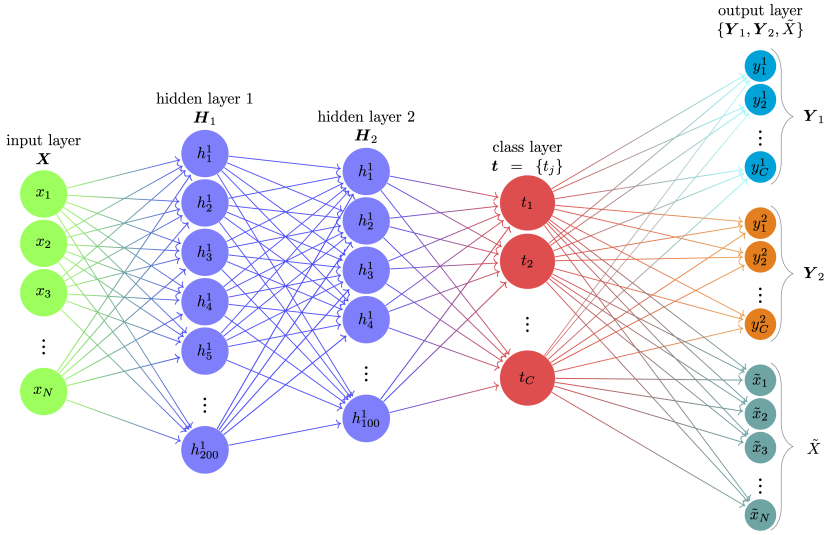

Figure 4.1: The structure of the proposed semi-supervised learning ANN The proposed semi-supervised learning ANN consisted of one input layer, two hidden layer, one class layer and one output layer.

added the second label as a regularization term encouraging the classification to be perceptually consistent. As mentioned in the previous session, in addition to DINA classification, we also choose the DINO classification as the second label concerning the two assumptions of the Co-Training method.

Let $\boldsymbol{x} \in \{0,1\}^I$ be the response patterns (I is the number of items), $\boldsymbol{c}_{DINA}$ and $\boldsymbol{c}_{DINO}$ be the One-Hot encoding of the DINA class labels and DINO class labels, respectively. Then we introduced into our ANN model the "true" latent class label (as opposed to the DINA and DINO class labels) as a latent multinomial variable $\boldsymbol{t} \in \{0,1\}^C$, $\sum_j^C t_j = 1$, where C is the number of latent classes. Like $\boldsymbol{c}_{DINA}$ and $\boldsymbol{c}_{DINO}$, $\boldsymbol{t}$ was also a One-Hot encoding label for each response pattern. The output of the third hidden layers (or the input of the output layer) of our ANN was the posterior over $\boldsymbol{t}$ using the softmax regression:

$$P(t_j = 1|\boldsymbol{x}) = \frac{\tilde{P}(t_j = 1|\boldsymbol{x})}{\sum_{j'=1}^{C} \tilde{P}(t_{j'} = 1|\boldsymbol{x})} = \frac{\phi_j(\boldsymbol{x})}{\sum_{j'=1}^{C} \phi_{j'}(\boldsymbol{x})} \tag{4.3}$$

where $\tilde{P}$ denotes the unnormalized probability distribution, $\Phi = \phi_j(\boldsymbol{x}), j \in \{1, \ldots, C\}$ indicates the calculation from the input layer to third hidden layer's output, and $\phi_j(\boldsymbol{x})$ indicates the jth node's values on the third hidden layer.

Given the true label $\boldsymbol{t}$, the output 1 (DINA model classification) and output 2 (DINO model classification) can be modeled using another softmax with logits as follows:

$$\begin{aligned} logit(P(c^{k}_{DINA} = 1|\boldsymbol{x})) &= \sum_{j=1}^{C} w_{kj} t_j \\ logit(P(c^{k'}_{DINO} = 1|\boldsymbol{x})) &= \sum_{j=1}^{C} w_{k'j} t_j \end{aligned} \tag{4.4}$$

where the w_{kj} and $w_{k'j}$ learn the log-probability of the "true" label j as DINA class label k (the kth class in DINA classification) and as DINO class label k' (the k'th class in DINA classification), respectively. Thus, in the proposed ANN, the joint relationship between input layer x and the kth node of output 1 and k'th node of output 2 can be represented as follows:

$$\begin{aligned} &P(c^{k}_{DINA} = 1, c^{k'}_{DINO} = 1|\boldsymbol{x}) = \\ &\sum_{j=1}^{C} P(c^{k}_{DINA} = 1, c^{k'}_{DINO} = 1, t_j = 1|\boldsymbol{x}) = \\ &\sum_{j=1}^{C} P(c^{k}_{DINA} = 1|t_j = 1) P(c^{k'}_{DINO} = 1|t_j = 1) P(t_j = 1|\boldsymbol{x}) \end{aligned} \tag{4.5}$$

where $(t_j = 1|\boldsymbol{x})$, $P(c^{k'}_{DINO} = 1|t_j = 1)$, $P(c^{k}_{DINA} = 1|t_j = 1)$ were defined in equation 4.3, and 4.4. Then, we could perform training via stochastic gradient descent on logit of the equation 4.5 to minimize the following cost function:

$$\{\boldsymbol{w}\} = \arg\min\{H(\boldsymbol{Y}_1, \boldsymbol{c}_{DINA}) + H(\boldsymbol{Y}_2, \boldsymbol{c}_{DINO})\} \tag{4.6}$$

where $\{\boldsymbol{w}\}$ indicates the parameters contained in the ANNs, H is the cross-entropy to calculate the difference between $\boldsymbol{Y}_1$ (output 1) and DINA labels $\boldsymbol{c}_{DINA}$, and the difference between $\boldsymbol{Y}_2$ (output 2) and DINA labels $\boldsymbol{c}_{DINO}$.

In addition to $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$, as what we did in the unsupervised learning framework, we added a regularization term, $H(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, into equation 4.6 to encourage the classification to be perceptually consistent. $\boldsymbol{x}$ is the observed response pattern, and $\tilde{\boldsymbol{x}}$ is the reconstructed response pattern corresponding to the estimated latent class. A general $I \times C$ item by latent class matrix Π (Xu & Zhang, 2016) was used to determine the conditional probability that students

in cth latent class answer ith item correctly $P(x_i = 1|c) = \pi_{i,c}$, and

$$\Pi = \begin{bmatrix} \pi_{1,1} & \pi_{1,2} & \cdots & \pi_{1,C} \\ \pi_{2,1} & \pi_{2,2} & \cdots & \pi_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{I,1} & \pi_{I,2} & \cdots & \pi_{I,C} \end{bmatrix}. \tag{4.7}$$

where I indicated the number of items, C indicated the number of latent classes. Then the reconstructed response pattern are calculated as following:

$$\tilde{\boldsymbol{x}} = \{x_i\} = \{\sum_{j=1}^{c} P(t_j = 1|\boldsymbol{x})\pi_{i,j}\} \tag{4.8}$$

After adding the regularization term, equation 4.6 can be represented as:

$$\{\boldsymbol{w}\} = \arg\min\{H(\boldsymbol{Y}_1, \boldsymbol{c}_{DINA}) + H(\boldsymbol{Y}_2, \boldsymbol{c}_{DINO}) + \lambda H(\boldsymbol{x}, \tilde{\boldsymbol{x}})\} \tag{4.9}$$

where λ is a scaling parameter which was determined through a validation test.

Because of the large number of parameters contained in the deep learning structure, the random initialization of parameters may impact the optimization when the training sample size is not large enough. Thus, one concern of using ANNs for CDM is that using the feature extracted by deep learning through a single training is risky or sensitive to the starting points of the parameters (Briggs & Circi, 2017). Cui et al. (2016) only set a maximum number of iterations (e.g., 10,000) to stop training the supervised learning ANN in their research study. We applied two methods to deal with this issue. The first method was the early stopping, which is a simple, effective, and widely used approach to avoid over-training the ANNs. The early stopping method is used to train on the training dataset but to stop training at the point when performance on a validation dataset starts to degrade. In addition, through the validating, we determined the scaling parameter in equation 10. In our method, the whole data set was divided into two parts: the training dataset consisted of 80% observations, and the validating dataset consisted of the rest 20% observations. The second method was that we conducted 100 ANN trainings individually, produced a probability of latent class for each training, and then averaged the 100 probabilities as the final probability of the latent class for each test-taker.

4.3 Experimental Study

The aims of the experiment were (1) to examine the attribute profile estimation and classification accuracy of the proposed method under different test factors which are expected to affect the estimates' accuracy, and (2) to compare the proposed method with the performance of five TDCMs: the DINA, DINO, G-DINA (De La Torre, 2011), LCDM (Henson et al., 2009), and RUM (Hartz, 2002). Thus, we conducted a simulation study under different assessment conditions with a variety of fixed factors and four manipulated factors.

4.3.1 Method

Manipulated Factors. Using item by latent class matrix, we manipulated three assessment factors in the data generation for the simulation, including the number of items (20 or 30), number of attributes (three or four), and test diagnostic quality (high or mixed). When estimating the conditions, we also manipulated the Q-matrix accuracy (100% and 90% correct) as another factor expected to impact classification accuracy.

Test length and number of attributes. The number of items (20 or 30) and the number of attributes were selected to reflect the current real assessment applications, which often contained between 20 to 30 items and measured three or four attributes (e.g., MELAB data, Li and Suen, 2013; DTMR data, Bradshaw et al., 2014). For three attributes, we generated 20 items, and for four attributes, 20 and 30 items were generated, respectively. The three Q-matrices (i.e., 20 items measured 3 attributes, 20 items measured 4 attributes, and 30 items measured 4 attributes) for these conditions are shown in the Appendix **??**, **??**, and **??**, respectively.

Test diagnostic quality. Item discriminating power is another factor impact performance of TDCMs shown in previous research studies (e.g., Cui et al., 2016; Roussos et al., 2005). The item discriminating power d_i is calculate as $d_i = p(x = 1|\alpha_1) - p(x = 1|\alpha_0)$. α_0 is the attribute pattern where none of the attributes measured by the ith item are mastered, and α_1 is the attribute pattern where all attributes measured by the ith item are mastered. If $.3 \geq d_i < .75$, the Item i is a highly discriminating item, and if $0 < d_i \leq .3$, the Item i is a lowly discriminating item. In the assessments with high diagnostic quality, all items are of high discriminating power; in the assessments with mixed diagnostic quality, 50% items are of high discriminating power, and 50% items are of low discriminating power.

Accuracy of Q-matrix. Since the Q-matrices constructed by content experts do not always reflect the relationship precisely and may require empirically-

driven modifications (Bradshaw et al., 2014; Tjoe & de la Torre, 2014) , two levels of Q-matrix accuracy were also created for TDCMs model fitting and Co-Training methods: 100% accuracy indicated that the Qmatrix were completely known; 90% accuracy indicated that 10% of elements in each Q-matrix were incorrect. We mis-specified the 10% elements in Q-matrix randomly drawing a Q-matrix entries and changing its value, with the constraint that each item must measure at least one attribute (i.e., a randomly drawn value of "1" for a simple structure item could not be changed to "0"). Such constrain makes there is no all zero q-vector (e.g., [0, 0, 0], [0, 0, 0, 0]) in Q-matrix.

Generating item response probabilities. Sample sizes of 1000 were used for all conditions. The true class probabilities of correct response for the items in the item pools were simulated using the logic of a DCM with respect to the Q-matrix defining the item-class relationships and the probabilities following monotonicity constraints across non-equivalence classes on an item (i.e., masters of all attributes measured by the item having a higher probability of correct response than masters of a proper subset of these attributes; masters of no attributes measured by the item having a lower probability of correct response than masters of a proper subset of these attributes), but did not follow a particular existing DCM item response function (e.g., the LCDM or DINA function). Current DCM item response functions constrain the item response probabilities to be equal within all equivalence classes; our simulated data did not. Item-based equivalence classes are latent classes that have the same attribute profile, or the same pattern of mastery, for all attributes that are measured by the item. Conversely, item-based nonequivalence classes differ on the mastery status of one or more attributes measured by the item.

We simulated data using a general $I \times C$ item by latent class matrix (Xu & Zhang, 2016) according to TDCM logic (i.e., defining latent classes by attribute profiles and specifying itemlatent class relationships by the Q-matrix) without the specific mathematic representation of the item response function:

$$\Pi = \begin{bmatrix} \pi_{1,1} & \pi_{1,2} & \dots & \pi_{1,C} \\ \pi_{2,1} & \pi_{2,2} & \dots & \pi_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{I,1} & \pi_{I,2} & \dots & \pi_{I,C} \end{bmatrix}. \tag{4.10}$$

where the conditional probability that students in lth latent class answer ith item correctly $P(x_i = 1|c) = \pi_{i,c}$, which is also known as item response probability (IRP) for each class. I indicated the number of items, C indicated the number of latent classes.

We denote π_{i,α_0}, π_{i,α_1}, and $\pi_{i,\alpha_{\boldsymbol{p}}}$ as the IRPs for non-mastery group, mastery group, and partial mastery group respectively. The mastery group contained students who mastered all of the attributes required by ith item, the partial mastery group contains students who only mastered a proper subset of attributes required by ith item, and the non-mastery group contained students who mastered none of the attributes required by ith item.

As shown in Table 4.1, when simulating response patterns to high discrimination items for the mastery group π_{i,α_1} were drawn from a uniform distribution $U[.65, .9]$; for the non-mastery group π_{i,α_0} were drawn from a uniform distribution $U[.15, .35]$; and for the partial mastery group $\pi_{i,\alpha_{\boldsymbol{p}}}$ were drawn from a uniform distribution $U[.4, .6]$. These draws yielded an average item discrimination value of .530 in 3 highly discriminating assessments (see values of IRP tables in Appendix **??**, **??** and **??**). When simulating response patterns to low discrimination items, for the non-mastery group π_{i,α_0} were drawn from a uniform distribution $U[.2, .4]$; for partial mastery group $\pi_{i,\alpha_{\boldsymbol{p}}}$ were drawn from a uniform distribution $U[\pi_{i,\alpha_0}, \pi_{i,\alpha_0} + .2]$; lastly for the mastery group (students who mastered all the attributes required by ith item) π_{i,α_1} were based on a uniform distribution $U[\pi_{i,\alpha_{\boldsymbol{p}}}, \pi_{i,\alpha_0} + .3]$ for complex items and $U[\pi_{i,\alpha_0}, \pi_{i,\alpha_0} + .3]$ for simple items. This yielded an average item discrimination value of .387 in 3 mixed discriminating assessments (see values of IRP tables in Appendix **??**, **??** and **??**).

Table 4.1: The table of selecting $\pi_{i,c}$ for item by class matrix.

Latent Groups	High Discrimination	Low Discrimination
Non-mastery π_{i,a_0}	$\mathbf{U}[.15, .35]$	$\mathbf{U}[.20, .40]$
Partial-mastery $\pi_{i,\alpha_{\boldsymbol{p}}}$	$\mathbf{U}[.40, .60]$	$\mathbf{U}[\pi_{i,a_0}, \pi_{i,a_0} + .15]$
Mastery π_{i,a_1}	$\mathbf{U}[.65, .90]$	$\begin{cases} U[\pi_{i,\alpha_{\boldsymbol{p}}}, \pi_{i,a_0} + .30], & \text{complex items} \\ U[\pi_{i,a_0}, \pi_{i,a_0} + .30], & \text{simple items} \end{cases}$

Note. For each item, π_{i,a_0}, π_{i,a_1}, and $\pi_{i,\alpha_{\boldsymbol{p}}}$ indicate the $\pi_{i,c}$ for non-mastery group, mastery group, and partial mastery group, respectively.

By drawing true item parameters in this way, the $\pi_{i,c}$s in our simulated data differs from IRPs simulated from the LCDM in that partial mastery classes with the same attribute pattern with respect to the measured attributes on a given item (the partial mastery item-based equivalence classes) have different true item response probabilities. The item response probabilities for these classes are, however, drawn from the same uniform distribution, so while they may be different values, they will be in the same range. Taking Item 10 that measures Attribute 1 and Attribute 2 as an example (as shown in Appendix **??**), Classes C2,

C3, C6 and C7 are all partial mastery classes with respect to this item: Class C2 and C6 both have mastered Attribute 1 but not Attribute 2, and Class C3 and C7 has both mastered Attribute 2 and not Attribute 1. Under the LCDM, Class C2 and C6 would have the same IRP, while Class C3 and C7 would have the same IRP; under our generating model, the IRP for all four classes were drawn from the same interval, but the draws were different, resulting in, Class C2 having an IRP of .509, Class C3 having an IRP of .519, Class C6 having an IRP of .458, and Class C7 having an IRP of .429 (see Appendix **??**). For non-mastery equivalence classes and mastery equivalence classes, the true model did constrain draws to be equal within the interval (i.e., Class C1 and C5 have IRP values of .33 and Class C4 and C8 have IRP values of .891). Only for partial mastery item-based equivalence classes were they allowed to differ. The purpose of allowing this difference was to add some noise in the data while still controlling the item discrimination level (IRP of mastery group minus IRP of non-mastery group).

The values in the item by latent class matrix Π for the 6 item pools are shown in Appendix **??**, **??**, **??**, **??**, **??**, and **??**, respectively. These appendices showed that the DCMs primary monotonicity assumptions held. Namely, the mastery group has the greatest IRP, the non-mastery group has the lowest IRP, and the IRP of partial mastery groups lie between them.

The IRPs for the non-mastery group and the mastery group are shown in Appendix D to I. These appendices show this simulation procedure firstly held that $.3 \leq d_i < .75$ for high discrimination items and $.3 < d_i < .75$ for low discrimination items; it also again shows the DCM monotonicity assumptions that the mastery group has a greater IRP than the non-mastery group held.

Estimation. In our simulated study, as a comparison, five types of widely used TDCMs were introduced as baselines to evaluate the diagnostic classification performance of the proposed framework. DINA and DINO models were selected as two baselines because they were the two classifiers used for Co-Training method. In addition, we chose three more general models, the G-DINA with identity link function (De La Torre, 2011), the LCDM with the logit link function (Henson et al., 2009), and the RUM (Hartz, 2002).

Results were analyzed in terms of classification accuracy of the five TDCMs and proposed method under 12 different test conditions. Since in the proposed method, a validation test was introduced for early stop in the training procedure to avoid overtraining, the whole data set was divided to two parts: training dataset which contains 80% observations; and validating dataset which contains 20% observations. In the results shown in Table 4.2, 4.3, and 4.4, we list three types of the results of using the proposed ANN method:

1. ANN: the classification results of applying the trained ANN structure to the whole dataset containing training set and validation set;

2. ANN*: the classification results of applying the trained ANN structure to the training dataset;

3. ANN**: the classification results of applying the trained ANN structure to the validating dataset.

The data simulation and five TDCMs were conducted using the "CDM" package (George et al., 2016) in R. The proposed semi-supervised learning ANN was conducted using the "tensorflow" library (Pang et al., 2020) in Python. In the experimental study, we conducted 100 replications. In each replication, new response patterns were created based on the fixed values in the item by latent class matrices in Appendices **??** to **??**.

4.3.2 Results

First, we tested the effects of the four assessment factors of test length, number of attributes, test diagnostic quality, and Q-matrix accuracy on the attribute profile and classification accuracy for the proposed method. Then we compared the proposed method to the five TDCMs, under 12 different test conditions. Results are given in Table 4.2, 4.3, and 4.4.

Classification Accuracy and Four Assessment Factors We first focus on results for the proposed method. As mentioned in the Estimation session, ANN, ANN* and ANN** in Table 4.2, 4.3, and 4.4 indicate the classification accuracy on whole dataset (including training set and validating set), the training set and validating set, respectively. Results show that the proposed method (ANN) works reasonably well and has classification accuracy values greater than 70% under 6 out of 12 assessment conditions (condition 1, 2, 3, 4, 9 and 10) when applying the trained ANN to the whole data set (i.e., ANN). Condition 1 to 4 are all four test conditions for the assessment measures 3 attributes using 20 items with either highly diagnostic quality or mixed diagnostic quality. Condition 9 and 10 are the two test conditions for assessment measures 4 attributes using 30 items with highly diagnostic quality. Results show classification accuracy increased in expected ways for the proposed method. Namely, average classification accuracy increases from .670 to .722 as test length increases from 20 to 30 for assessments measure 4 attributes (there is only one test length of assessment that measures 3 attributes); when the number of attribute measured

decreases from 4 to 3 in assessment with 20 items, the average classification accuracy increases from .670 to .834; when the test diagnostic quality increases from mixed to high, the average classification accuracy increases from .621 to .736; and when the accuracy of Q-matrix increases from 90% to 100%, the average accuracy increases slightly from .675 to .682. In addition, we can see that ANN* always achieves the best performance with average classification accuracy .692, ANN** always achieves the worst performance with average classification accuracy . 661, and ANN falls between ANN* and ANN** with average classification accuracy .678. The reason is that the parameters of ANN structure were trained based on the training dataset but not considered the validation dataset.

Next, we examine the results for the five TDCMs. Results show that DINA model has classification accuracy values greater than 70% under 2 out of 12 assessment conditions (condition 1 and 2); DINO model has classification accuracy values greater than 70% under 2 out of 12 assessment conditions (condition 1 and 2); G-DINA has classification accuracy values greater than 70% under 5 out of 12 test conditions (condition 1, 2, 3, 9, and 10); LCDM has classification accuracy values greater than 70% under 5 out of 12 test conditions (condition 1, 2, 3, 9, and 10); and RUM has classification accuracy values greater than 70% under 5 out of 12 test conditions (condition 1, 2, 3, 9, and 10). Condition 1 and 2 are two tests (high and mixed diagnostic quality) with 20 items measures 3 attributes and the Q-matrix accuracy is 100%; condition 3 is a test with high diagnostic quality consists of 20 items to measure 3 attribute but the Q-matrix accuracy is 90%; condition 9 and 10 are two tests (high and mixed diagnostic quality) with 30 items measures 4 attributes and the Q-matrix accuracy is 100%. We could also notice that the G-DINA and LCDM achieved almost the same classification results because the only difference between G-DINA and LCDM in the CDM::gdina() is the link function. We chose 'identity' function for G-DINA and 'logit' function for LCDM. In addition, like the proposed method, results show classification accuracy increased in expected way for the 5 TDCMs. Namely, accuracy increases as test length increases, as the number of attribute measured decreases, as the test diagnostic quality increases, and as the accuracy of Q-matrix increases.

Comparison Classification with 5 TDCMs Simulation results indicated that when using the proposed ANN, the classification rates were higher than rates from the DINA and DINO models, the two initial classifiers used in Co-Training. Compared to DINA and DINO models, at the attribute level, the average improvements of classification using ANN was .0218 and .0140, and at the class level (i.e., attribute profiles level), the average improvements were .0589

and .0432. Compared to the general models LCDM and G-DINA, which often achieved the best performance in classification, the performance of ANN was also better than these two methods. The improvements at the attribute level were .0056 and .0055 compared with LCDM and G-DINA models, respectively. At the class level, the improvements were .0130 and .0132.

The simulated study also indicated that when the Q-matrix became less accurate, the classification accuracy for each method dropped at both attribute level and latent class level when holding other test assessment factors. When the Q-matrix accuracy decreased to 90% accurate, at the attribute level, the average reductions of classification accuracy were .0071, .0055, .0114, .0114, .0095, and .0038 corresponding to DINA, DINO, LCDM, G-DINA, RUM, and our ANN methods respectively. At the attribute pattern level, the average accuracy reductions were .0163, .0138, .0298, .0302, .0243, and .0075 for DINA, DINO, LCDM, G-DINA, RUM and, our ANN methods respectively. From this observation, we could find that firstly, the relaxed models (LCDM, G-DINA, and RUM) were more sensitive to the accuracy of Q-matrix; secondly, the proposed ANN was more robust to the noise within the Q-matrix compared to the five TDCMs.

Besides, high item discriminating was a positive impact on the classification accuracy of all six methods. When the discrimination of items decreased (from high to mixed), the classification rate dropped .0301, .0383, .0458, .0458, .0392, and .0397 for DINA, DINO, LCDM, G-DINA, RUM and our ANN at the attribute level. The reductions were .0780, .1095, .1318, .1318, .1137 and .1158 for DINA, DINO, LCDM, G-DINA, RUM, and our ANN at the latent class level. The reason that our ANN method dropped more than DINA, DINO, and RUM (only at the attribute level) was that when the items were high discriminating, the improvement of classification rate using our ANN was more significant than using mixed discriminating items. Even though the performance of our ANN at both the attribute level and the latent class level was the best among the six diagnostic classification methods.

4.4 Conclusion

The purpose of this research is to solve two problems that exist in current supervised learning ANN methods and unsupervised learning ANNs: the supervised learning method requires ideal response pattern to train the model; the classification accuracy of unsupervised learning methods was not as good as TDCMs. We designed a novel semi-supervised learning ANN to do diagnostic classification and evaluated the performances of the proposed method through

a simulation study. In the proposed framework, we combined ANN with a semi-supervised learning method, the Co-Training method. To hold the two assumptions of successfully applying Co-Training, we used two TDCMs, DINA and DINO models, as the two classifiers.

In the simulated study, we compared the proposed method with five widely used TDCMs, DINA, DINO, LCDM, G-DINA, and RUM. By varying the four assessment factors (item discrimination, Q-matrix accuracy, number of attributes and items) which impact the performance of TDCMs, the comparison results indicated some advantages of the proposed method.

The first advantage is that the proposed ANN method achieved comparable performance compared with the five TDCMs even under the ideal assessment condition (high diagnostic quality and 100% Q-matrix accuracy). It means that the proposed ANN method could be used for providing reasonable cognitive diagnostic classification result without an appropriate TDCM for an assessment.

The second advantage is that proposed ANN was robust to the Q-matrix misspecification because the classification rate dropped less than the other five TDCMs when the Q-matrix accuracy decreased to 90% accuracy. This advantage make the proposed method can be used for real large scale assessment because the Q-matrix of a large number of items can hardly be guaranteed to be 100% accurate.

The last advantage is that although the classification rates of the proposed method dropped more than DINA and DINO when the item discriminating power reduced, the proposed method was still more robust to the item discriminating reduction than the general TDCMs. In other words, the proposed method finds a trade-off between classification accuracy and robustness to the noise.

Generally, the proposed method could demonstrated the ability to provide a reasonably accurate classification results which can be used for either providing diagnostic classification. In addition, the classification can be used to determine the relationship between items and latent class. Then, the relationship can help researchers to choose the appropriate TDCM to fit the data and estimate both personal variable and item variables.

4.5 Discussion

Although the study demonstrates promise for using the proposed semi-supervised learning artificial neural networks, there are still some limitations. One concern of this study is that the current analysis only focused on the classification rate but not consider the item parameters, which are very important to provide ap-

propriate item matching students' ability in an computer adaptive test or online adaptive learning environment. Another concern of this study is that the missing response was not considered in the proposed ANN. In the simulation, we assumed that all test-takers responded all items, but in real assessment, the missing ness is a very common issue in CDM. The last concern is that although we introduced the validating test for early stop to avoid over training, this research did not evaluate the prediction performance of the proposed method. The reason is that in current CDM area, the research studies focus on explaining data not doing prediction on a new dataset. With regard to these three concerns, there will be three future research topics.

The first future study is that the classification results could be used to determine the item parameters to evaluate item discriminating power among students' mastery level for specific attributes or determine the relationship between items and attributes to explore the attribute structures. An appropriate difficulty that matches a student's momentary attribute profile is expected to encourage the student to complete the item.

The second future research direction is to convert the dichotomous response patterns to polychotomous response patterns by considering missing values into the input response pattern. Then a multiclass classification algorithm is applied to classify the latent classes by considering the missing values even the missingness is related to the latent class (i.e., non-ignorable missingness).

The last future research is to evaluate the prediction performance of the artificial neural network based cognitive diagnostic classification method, and compare the performance with the TDCMs in doing prediction on new dataset. With regard to the knowledge in educational data mining (EDM), the prediction will consist of two directions: (1) how is the model's performance on predicting new test-takers' latent variables; (2) how is the model's performance on estimating new item's characteristics. For different directions, the ANN based method will be built up using different architecture.

Table 4.2: Comparison of classification rates for 3 attributes using 20 items.

Test Condition	Methods	Quality	Q-matrix Accuracy	Attribute 1	Attribute 2	Attribute 3	Class
1	DINA	High	100%	.949 (.00)	.864 (.02)	.957 (.01)	.778 (.02)
	DINO			.953 (.01)	.871 (.02)	.952 (.02)	.784 (.04)
	LCDM			.96 (.00)	.917 (.00)	.957 (.00)	.842 (.01)
	G-DINA			.96 (.00)	.917 (.00)	.957 (.01)	.842 (.00)
	RUM			.953 (.01)	.91 (.00)	.958 (.00)	.827 (.00)
	ANN			.956 (.01)	.915 (.01)	.957 (.01)	.834 (.02)
	ANN*			.962 (.01)	.921 (.01)	.964 (.02)	.851 (.02)
	ANN**			.945 (.01)	.901 (.02)	.942 (.01)	.818 (.03)
2	DINA		90%	.944 (.00)	.824 (.01)	.957 (.00)	.741 (.02)
	DINO			.946 (.01)	.852 (.01)	.944 (.01)	.757 (.02)
	LCDM			.956 (.00)	.897 (.00)	.958 (.00)	.819 (.00)
	G-DINA			.956 (.00)	.897 (.00)	.958 (.01)	.819 (.00)
	RUM			.949 (.00)	.879 (.01)	.958 (.00)	.794 (.01)
	ANN			.955 (.01)	.900 (.02)	.958 (.02)	.821 (.02)
	ANN*			.962 (.01)	.910 (.02)	.969 (.02)	.831 (.03)
	ANN**			.945 (.01)	.881 (.02)	.932 (.04)	.807 (.04)
3	DINA	Mixed	100%	.875 (.00)	.859 (.01)	.914 (.00)	.693 (.01)
	DINO			.863 (.01)	.864 (.00)	.896 (.01)	.665 (.01)
	LCDM			.879 (.01)	.884 (.00)	.913 (.00)	.712 (.01)
	G-DINA			.879 (.00)	.884 (.00)	.913 (.00)	.712 (.00)
	RUM			.873 (.01)	.9 (.00)	.917 (.01)	.724 (.00)
	ANN			.883 (.01)	.884 (.02)	.915 (.01)	.720 (.01)
	ANN*			.892 (.01)	.896 (.01)	.929 (.02)	.730 (.02)
	ANN**			.868 (.01)	.878 (.02)	.911 (.01)	.704 (.02)
4	DINA		90%	.878 (.01)	.85 (.01)	.906 (.00)	.676 (.02)
	DINO			.869 (.00)	.861 (.00)	.908 (.00)	.679 (.01)
	LCDM			.878 (.00)	.85 (.00)	.918 (.00)	.685 (.01)
	G-DINA			.877 (.00)	.85 (.01)	.918 (.00)	.684 (.00)
	RUM			.877 (.00)	.85 (.01)	.915 (.00)	.685 (.01)
	ANN			.874 (.01)	.888 (.02)	.908 (.02)	.704 (.02)
	ANN*			.889 (.01)	.901 (.01)	.923 (.01)	.719 (.01)
	ANN**			.867 (.04)	.871 (.04)	.890 (.03)	.683 (.03)

Note. ANN indicate the attribute profile estimation using the proposed method on whole data set; ANN* indicate the attribute profile estimation using the proposed method on the training data set; ANN** indicate the attribute profile estimation using the proposed method on the validation data set.

Table 4.3: Comparison of classification rates for 4 attributes using 20 items.

Test Condition	Methods	Quality	Q-matrix Accuracy	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Class
5	DINA	High	100%	.908 (.02)	.924 (.03)	.79 (.02)	.893 (.02)	.591 (.03)
	DINO			.909 (.04)	.928 (.05)	.858 (.02)	.899 (.03)	.653 (.04)
	LCDM			.918 (.01)	.929 (.02)	.858 (.00)	.919 (.01)	.67 (.01)
	G-DINA			.918 (.01)	.929 (.01)	.858 (.01)	.919 (.00)	.67 (.01)
	RUM			.923 (.02)	.921(.01)	.853 (.02)	.917 (.01)	.664 (.03)
	ANN			.919 (.01)	.925 (.01)	.858 (.03)	.922 (.04)	.67 (.03)
	ANN*			.931 (.02)	.942 (.01)	.870 (.03)	.941 (.01)	.691 (.03)
	ANN**			.909 (.03)	.918 (.02)	.861 (.01)	.912 (.04)	.655 (.03)
6	DINA		90%	.909 (.04)	.922 (.04)	.74 (.02)	.886 (.02)	.56 (.03)
	DINO			.903 (.04)	.924 (.02)	.852 (.03)	.879 (.04)	.621 (.04)
	LCDM			.904 (.01)	.922 (.00)	.824 (.01)	.887 (.01)	.616 (.01)
	G-DINA			.904 (.01)	.922 (.01)	.824 (.01)	.887 (.02)	.616 (.01)
	RUM			.905 (.02)	.922 (.02)	.8 (.02)	.884 (.01)	.599 (.03)
	ANN			.912 (.04)	.923 (.01)	.862 (.03)	.89 (.02)	.648 (.03)
	ANN*			.924 (.02)	.931 (.01)	.877 (.01)	.901 (.02)	.657 (.02)
	ANN**			.903 (.01)	.917 (.02)	.853 (.03)	.883 (.02)	.632 (.03)
7	DINA	Mixed	100%	.854 (.01)	.836 (.03)	.824 (.02)	.851 (.03)	.503 (.02)
	DINO			.863 (.02)	.817 (.04)	.854 (.02)	.816 (.04)	.484 (.04)
	LCDM			.867 (.01)	.823 (.01)	.855 (.02)	.84 (.03)	.509 (.01)
	G-DINA			.867 (.01)	.824 (.02)	.855 (.04)	.84 (.03)	.51 (.01)
	RUM			.878 (.03)	.831 (.02)	.856 (.03)	.837 (.04)	.522 (.03)
	ANN			.864 (.04)	.842 (.02)	.857 (.03)	.859 (.02)	.531 (.02)
	ANN*			.879 (.01)	.855 (.03)	.870 (.02)	.871 (.01)	.550 (.02)
	ANN**			.853 (.03)	.839 (.02)	.826 (.02)	.850 (.04)	.504 (.05)
8	DINA		90%	.856 (.04)	.826 (.02)	.744 (.01)	.854 (.01)	.448 (.02)
	DINO			.854 (.02)	.817 (.02)	.855 (.01)	.851 (.04)	.503 (.05)
	LCDM			.865 (.00)	.817 (.01)	.776 (.02)	.844 (.01)	.469 (.01)
	G-DINA			.865 (.02)	.817 (.01)	.776 (.00)	.844 (.01)	.469 (.01)
	RUM			.864 (.03)	.821 (.01)	.855 (.04)	.84 (.01)	.509 (.03)
	ANN			.852 (.02)	.871 (.02)	.855 (.03)	.852 (.01)	.542 (.04)
	ANN*			.869 (.03)	.883 (.02)	.867 (.01)	.870 (.00)	.558 (.03)
	ANN**			.850 (.04)	.851 (.02)	.855 (.02)	.854 (.03)	.512 (.05)

Note. ANN indicate the attribute profile estimation using the proposed method on whole data set; ANN* indicate the attribute profile estimation using the proposed method on the training data set; ANN** indicate the attribute profile estimation using the proposed method on the validation data set.

Table 4.4: Comparison of classification rates for 4 attributes using 30 items.

Test Condition	Methods	Quality	Q-matrix Accuracy	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Class
9	DINA	High	100%	.937 (.04)	.938 (.03)	.814 (.06)	.892 (.02)	.641 (.05)
	DINO			.942 (.01)	.941 (.08)	.854 (.11)	.902 (.08)	.681 (.12)
	LCDM			.947 (.01)	.949 (.00)	.873 (.02)	.925 (.02)	.732 (.02)
	G-DINA			.947 (.00)	.949 (.01)	.873 (.02)	.925 (.03)	.732 (.02)
	RUM			.948 (.02)	.945 (.00)	.872 (.04)	.917 (.03)	.719 (.05)
	ANN			.949 (.01)	.944 (.02)	.872 (.03)	.916 (.02)	.722 (.03)
	ANN*			.955 (.01)	.952 (.01)	.880 (.02)	.935 (.01)	.741 (.02)
	ANN**			.942 (.02)	.940 (.02)	.860 (.04)	.903 (.03)	.711 (.04)
10	DINA		90%	.934 (.03)	.94 (.02)	.853 (.04)	.853 (.03)	.64 (.04)
	DINO			.935 (.02)	.924 (.03)	.855 (.08)	.874 (.03)	.644 (.06)
	LCDM			.948 (.00)	.946 (.01)	.858 (.02)	.92 (.03)	.708 (.04)
	G-DINA			.948 (.02)	.946 (.01)	.859 (.03)	.92 (.03)	.709 (.03)
	RUM			.945 (.01)	.945 (.01)	.869 (.02)	.915 (.01)	.713 (.03)
	ANN			.952 (.02)	.948 (.01)	.873 (.02)	.916 (.01)	.723 (.02)
	ANN*			.960 (.02)	.954 (.02)	.890 (.01)	.926 (.01)	.733 (.02)
	ANN**			.935 (.03)	.940 (.04)	.860 (.02)	.902 (.04)	.703 (.04)
11	DINA	Mixed	100%	.903 (.03)	.876 (.03)	.801 (.01)	.882 (.02)	.56 (.02)
	DINO			.911 (.03)	.884 (.05)	.858 (.06)	.858 (.04)	.586 (.07)
	LCDM			.912 (.03)	.886 (.02)	.857 (.02)	.88 (.02)	.616 (.03)
	G-DINA			.912 (.02)	.886 (.01)	.858 (.02)	.88 (.01)	.617 (.02)
	RUM			.9 (.02)	.884 (.01)	.858 (.02)	.871 (.03)	.592 (.03)
	ANN			.91 (.01)	.889 (.02)	.862 (.01)	.881 (.01)	.616 (.02)
	ANN*			.916 (.02)	.898 (.01)	.869 (.02)	.900 (.01)	.623 (.02)
	ANN**			.905 (.02)	.881 (.03)	.850 (.02)	.881 (.03)	.605 (.03)
12	DINA		90%	.908 (.03)	.887 (.03)	.847 (.01)	.876 (.03)	.603 (.02)
	DINO			.906 (.03)	.883 (.07)	.852 (.08)	.836 (.07)	.566 (.09)
	LCDM			.908 (.02)	.891 (.01)	.863 (.03)	.868 (.01)	.605 (.02)
	G-DINA			.908 (.01)	.891 (.02)	.863 (.03)	.868 (.01)	.605 (.02)
	RUM			.905 (.02)	.891 (.01)	.864 (.03)	.861 (.03)	.602 (.03)
	ANN			.909 (.01)	.885 (.02)	.859 (.01)	.871 (.02)	.61 (.02)
	ANN*			.921 (.01)	.903 (.02)	.869 (.01)	.878 (.01)	.624 (.01)
	ANN**			.901 (.03)	.889 (.02)	.850 (.01)	.857 (.03)	.603 (.03)

Note. ANN indicate the attribute profile estimation using the proposed method on whole data set; ANN* indicate the attribute profile estimation using the proposed method on the training data set; ANN** indicate the attribute profile estimation using the proposed method on the validation data set.

Chapter 5

Conclusion and Discussion

As a new research area, Artificial Neural Networks (ANNs) begins to attract some research attention for cognitive diagnostic classification. However, some research studies showed that there are some challenges of applying ANN to CDM: (1) the current supervised leaning ANNs methods for CDM required assumption of the item response function; (2) the unsupervised learning ANNs method for CDM requires further data analysis; (3) the ANNs methods produced very unstable and unappreciated estimation unless a great deal of care was taken to conduct sensitivity analyses. To solve these current disadvantages, in my dissertation, we tried to explore the feasibility of combining ANNs with TDCMs under both unsupervised learning and semi-supervised learning framework. In particular, this dissertation is comprised of two separate but related applications. The first application is an unsupervised learning ANNs for CDM and Q-matrix reconstruction; the second application is a semi-supervised learning ANNs combining with TDCMs to improve the classification accuracy.

5.1 Application 1: An Unsupervised Learning Artificial Neural Network for Cognitive Diagnostic Measurement

The research aim was to propose an unsupervised learning ANN with fewer constraints according to two potential issues in model-based cognitive diagnosis: model selection and Q-matrix misspecification. To achieve this target, we firstly designed an unsupervised learning ANN for attribute estimation which did not rely on a specific assumption of a selected item response function and only

requires partial Q-matrix information (i.e., accurate simple items' q-vectors); secondly, we proposed a Q-matrix reconstruction method using K-means clustering algorithms to correct or reconstruct the mis-specified or missing elements of the Q-matrix. We tested our methodology and compared it with two theoretical DCMs (DINA and LCDM) under 24 types of simulated test conditions according to test length, number of simple items per attribute, and the diagnostic quality of the test. The results showed that our methodology provided an option for users to analyze large scale assessment responses data when lacking prior knowledge about the items, and also could reconstruct the item-attribute relationship (Q-matrix) only using limited information. Another advantage of this methodology showed in experimental results is that unlike the typical DCMs which removes the items with low discriminating power or the items with unknown q-vector, the proposed method did not remove such items because the MAEN structure in the proposed method had the ability to explore useful information from these items which could not be used in typical DCMs.

But there are still some limits to our method. First, this method cannot determine the number of attributes. Second, as shown in experimental results, different types of test conditions, such as test length, simple item proportion, and diagnostic quality of a test, affect the estimation results. The last limit is when the number of items is small, the theoretical DCMs using partial Q-matrix provided better performance than the proposed one.

5.2 Application 2: Semi-supervised deep Co-Training method for CDM

The purpose of this application is to solve two problems that exist in current supervised learning ANN methods and unsupervised learning ANNs: the supervised learning method requires ideal response pattern to train the model; the classification accuracy of unsupervised learning methods was not as good as TDCMs.. We designed a novel semi-supervised learning ANN to do diagnostic classification and evaluated the performances of the proposed method through a simulation study. In the proposed framework, we combined ANN with a semi-supervised learning method, the Co-Training method. To hold the two assumptions of successfully applying Co-Training, we used two TDCMs, DINA and DINO models, as the two classifiers.

In the simulated study, we compared the proposed method with five widely used TDCMs, DINA, DINO, LCDM, G-DINA, and RUM. By varying the four assessment factors (item discrimination, Q-matrix accuracy, number of attributes and items) which impact the performance of TDCMs, the compar-

ison results indicated that the proposed ANN method achieved comparable performance compared with the five TDCMs even under the ideal assessment condition (high diagnostic quality and 100% Q-matrix accuracy). Also, the proposed ANN was robust to the Q-matrix misspecification because the classification rate dropped less than the other five TDCMs when the Q-matrix accuracy decreased to 90% accuracy. Although the classification rates of the proposed method dropped more than DINA and DINO when the item discriminating power reduced, the proposed method was still more robust to the item discrimination reduction than the general TDCMs. In other words, the proposed method finds a trade-off between classification accuracy and robustness to the noise. The classification results could also be useful to update the item response functions of TDCMs when giving a new assessment dataset.

One concern of this study is that the current analysis only focused on the classification rate of the proposed method. In the future study, the classification results could be used to analyze item parameters to evaluate item discriminating power among students' mastery level for specific attributes or determine the relationship between items and attributes to explore the attribute structures. Another concern of this study is that the missing response was not considered in the proposed ANN.

5.3 Exploration of Using ANNs for Diagnostic Classification

Together, these two studies show that ANN are a feasible method for cognitive diagnostic modeling. Both unsupervised learning ANN method and semi-supervised learning ANN method could provide reasonable classification accuracy. The classification accuracy of unsupervised learning ANN could be used to determine the relationship between item and attribute and determine which TDCM is appropriate for the assessment data. The semi-supervised learning ANN could provide appreciable classification accuracy as the TDCMs.

In addition, since ANN is a kinds of machine learning method which could explore useful information from observation, the proposed ANN based methods are more robust to the noises contained in the assessment, such as items with low discriminative power, or inaccurate elements in Q-matrix. It provides ANN based advantages to explore useful information that might be ignored by human from the observations.

5.4 Future Directions

With regard to the limitation of applying ANNs in these two applications, the future research direction can be divided into two parts: (1) explore the number of attributes for CDM; (2) consider missing responses in the ANNs; (3) integrate data analyzing for future psychometric application.

The first future research topic would be an investigation into a determination on the number of attributes before doing cognitive diagnostic classification. In the applications of this research, both unsupervised learning and semi-supervised learning ANNs for CDM held the assumption that the number of attributes and latent traits is known. Thus, the structures of these two kinds of ANNs cannot be directly built up for new assessment without the number of attributes.

The second future research topic would be a combination of a multiclass classification with the ANNs architecture. The multiclass classification considers the missing values as the third response type other than the correct response and incorrect response. The advantage of this combination is that all students' responses (including missingness) could be used in model training and parameter estimation. In assessment and online learning environment, the missingness may related to the latent variables (e.g., latent class, item parameters) measured by CDM. Considering the missingness into the ANNs architecture will increase the estimation accuracy and help to find out the relationship between missing values and the latent variables.

The last future research topic would be introducing new machine learning techniques into the ANNs to help improve the research in psychometrics. Transfer learning (TL; Pan and Yang, 2009) and item characteristic prediction using natural language processing (NLP; Manning et al., 1999) are potentially applicable. TL could provide a better start point and initialization of the parameters for training and decreasing the requirement of training sample size by using the information we have achieved from the previous items with a large number of responses. NLP could help to initialize the item parameters before embedding the question in real exam as unscored items. Given the increased assessment data, the generative and discriminative machine learning and deep learning models are more efficient for psychometric applications.

Appendix A

Π matrix of Item pool 1

Item	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
1	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***
2	.327*	.327*	.752***	.752***	.327*	.327*	.752***	.752***
3	.159*	.159*	.159*	.159*	.885***	.885***	.885***	.885***
4	.328*	.782***	.328*	.782***	.328*	.782***	.328*	.782***
5	.241*	.241*	.788***	.788***	.241*	.241*	.788***	.788***
6	.241*	.241*	.241*	.241*	.889***	.889***	.889***	.889***
7	.158*	.469**	.158*	.531**	.464**	.712***	.438**	.712***
8	.349*	.349*	.556	.419	.493	.502	.81***	.81***
9	.179*	.52**	.467**	.722***	.179*	.498**	.591**	.722***
10	.246*	.246*	.497**	.578**	.583**	.522**	.656***	.656***
11	.233*	.482**	.429**	.686***	.233*	.587**	.46**	.686***
12	.243*	.412**	.243*	.59**	.544**	.708***	.428**	.708***
13	.174*	.51**	.591**	.85***	.174*	.517**	.481**	.85***
14	.225*	.225*	.53**	.464**	.462**	.444**	.874***	.874***
15	.24*	.474**	.24*	.597**	.431**	.854***	.418**	.854***
16	.276*	.428**	.538**	.839***	.276*	.524**	.578**	.839***
17	.273*	.273*	.535**	.547**	.504**	.532**	.745***	.745***
18	.308*	.564**	.308*	.557**	.596**	.754***	.488**	.754***
19	.281*	.462**	.482**	.402**	.437**	.569**	.446**	.683***
20	.281*	.736***	.281*	.736***	.281*	.736***	.281*	.736***
21	.188*	.188*	.73***	.73***	.188*	.188*	.73***	.73***
22	.169*	.169*	.169*	.169*	.846***	.846***	.846***	.846***
23	.341*	.341*	.448**	.415**	.449**	.546**	.772***	.772***
24	.179*	.569**	.179*	.5**	.478	.753***	.449**	.753***
25	.178*	.422**	.478**	.83***	.178*	.514**	.443**	.83***
26	.214*	.489**	.214*	.444**	.5**	.812***	.471**	.812***
27	.194*	.194*	.194*	.194*	.727***	.727***	.727***	.727***
28	.347*	.347*	.742***	.742***	.347*	.347*	.742***	.742***
29	.328*	.53**	.475**	.805***	.328*	.471**	.507**	.805***
30	.297*	.818***	.297*	.818***	.297*	.818***	.297*	.818***

Note. * indicates the $\pi_{i,c}$ for non mastery group, ** indicates the $\pi_{i,c}$ for partial mastery group, *** indicates the $\pi_{i,c}$ for mastery group.

Appendix B

IRP[I] of Item Pool 1

Item	IRP Non-Mastery Group	IRP Mastery Group	Discrimination
1	.308	.722	.414
2	.327	.752	.426
3	.159	.885	.726
4	.328	.782	.454
5	.241	.788	.547
6	.241	.889	.649
7	.158	.712	.553
8	.349	.810	.461
9	.179	.722	.543
10	.246	.656	.411
11	.233	.686	.453
12	.243	.708	.465
13	.174	.850	.675
14	.225	.874	.649
15	.24	.854	.614
16	.276	.839	.563
17	.273	.745	.472
18	.308	.754	.447
19	.281	.683	.402
20	.281	.736	.455
21	.188	.730	.543
22	.169	.846	.677
23	.341	.772	.431
24	.179	.753	.573
25	.178	.830	.652
26	.214	.812	.598
27	.194	.727	.533
28	.347	.742	.396
29	.328	.805	.477
30	.297	.818	.521

Note. 1. IRP indicates the item response probability, also known as correct response rate.

Appendix C

True Values of λs^I under the LCDM for Item Pool 1

Item	λ_0 (Intercepts)	λ_1	λ_2	λ_3	λ_{12}	λ_{13}	λ_{23}	λ_{123}
1	-.811	1.765	0	0	0	0	0	0
2	-.724	0	1.834	0	0	0	0	0
3	-1.665	0	0	3.707	0	0	0	0
4	-.715	1.993	0	0	0	0	0	0
5	-1.145	0	2.458	0	0	0	0	0
6	-1.150	0	0	3.232	0	0	0	0
7	-1.670	1.770	0	1.671	0	-.869	0	0
8	-.624	0	.739	.859	0	0	.478	0
9	-1.520	1.600	1.642	0	-.766	0	0	0
10	-1.122	0	1.472	1.318	0	0	-1.021	0
11	-1.192	1.223	.970	0	-.221	0	0	0
12	-1.135	1.048	0	.850	0	.124	0	0
13	-1.555	1.277	1.676	0	.334	0	0	0
14	-1.237	0	1.144	1.189	0	0	.839	0
15	-1.154	1.057	0	1.017	0	.844	0	0
16	-.965	1.208	1.067	0	.338	0	0	0
17	-.982	0	.864	.859	0	0	.331	0
18	-.811	.595	0	.776	0	.563	0	0
19	-.941	.621	.889	1.259	-.176	-.626	-1.468	1.209
20	-.938	1.962	0	0	0	0	0	0
21	-1.466	0	2.461	0	0	0	0	0
22	-1.595	0	0	3.295	0	0	0	0
23	-.659	0	.651	.632	0	0	.597	0
24	-1.520	1.671	0	1.731	0	-.769	0	0
25	-1.527	1.525	1.628	0	-.040	0	0	0
26	-1.301	1.504	0	1.297	0	-.038	0	0
27	-1.425	0	0	2.404	0	0	0	0
28	-.633	0	1.691	0	0	0	0	0
29	-.716	.433	.627	0	1.073	0	0	0
30	-.860	2.364	0	0	0	0	0	0

Note. 1. λ_0 is intercept; λ_1, λ_2, and λ_3 are main effects; λ_{12}, λ_{13}, λ_{23}, and λ_{123} are interaction effects..

Appendix D

Π matrix of Item pool 2

Item	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
1	.304*	.502***	.304*	.502***	.304*	.502***	.304*	.502***
2	.364*	.364*	.6***	.6***	.364*	.364*	.6***	.6***
3	.396*	.396*	.396*	.396*	.528***	.528***	.528***	.528***
4	.262*	.385***	.262*	.385***	.262*	.385***	.262*	.385***
5	.202*	.202*	.257***	.257***	.202*	.202*	.257***	.257***
6	.369*	.369*	.369*	.369*	.438***	.438***	.438***	.438***
7	.248*	.359**	.248*	.281**	.31**	.461***	.288**	.461***
8	.278*	.278*	.372**	.306**	.408**	.39**	.41***	.41***
9	.289*	.389**	.382**	.462***	.289*	.345**	.368**	.462***
10	.271*	.271*	.402**	.358**	.397**	.318**	.432***	.432***
11	.326*	.432**	.366**	.561***	.326*	.415**	.398**	.561***
12	.274*	.314**	.274*	.359**	.411**	.458***	.409**	.458***
13	.342*	.383**	.39**	.55***	.342*	.49**	.435**	.55***
14	.383*	.383*	.524**	.453**	.444**	.482**	.625***	.625***
15	.387*	.41**	.387*	.473**	.423**	.602***	.531**	.602***
16	.248*	.338**	.325**	.529***	.248*	.308**	.38**	.529***
17	.273*	.273*	.328**	.316**	.299**	.299**	.403***	.403***
18	.243*	.315**	.243*	.281**	.275**	.428***	.344**	.428***
19	.364*	.371**	.469**	.417**	.425**	.487**	.502**	.57***
20	.296*	.464***	.296*	.464***	.296*	.464***	.296*	.464***
21	.34*	.34*	.614***	.614***	.34*	.34*	.614***	.614***
22	.324*	.324*	.324*	.324*	.452***	.452***	.452***	.452***
23	.308*	.308*	.35**	.452**	.417**	.411**	.568***	.568***
24	.231*	.239**	.231*	.29**	.303**	.465***	.315**	.465***
25	.262*	.367**	.399**	.52***	.262*	.355**	.326**	.52***
26	.251*	.332**	.251*	.26**	.29**	.506***	.311**	.506***
27	.234*	.234*	.234*	.234*	.355***	.355***	.355***	.355***
28	.294*	.294*	.555***	.555***	.294*	.294*	.555***	.555***
29	.385*	.415**	.51**	.618***	.385*	.408**	.506**	.618***
30	.267*	.371***	.267*	.371***	.267*	.371***	.267*	.371***

Note. * indicates the $\pi_{i,c}$ for non mastery group, ** indicates the $\pi_{i,c}$ for partial mastery group, *** indicates the $\pi_{i,c}$ for mastery group.

Appendix E

IRP[I] of Item Pool 2

Item	IRP Non-Mastery Group	IRP Mastery Group	Discrimination
1	.304	.502	.198
2	.364	.600	.236
3	.396	.528	.132
4	.262	.385	.123
5	.202	.257	.055
6	.369	.438	.069
7	.248	.461	.213
8	.278	.410	.132
9	.289	.462	.173
10	.271	.432	.161
11	.326	.561	.235
12	.274	.458	.184
13	.342	.550	.208
14	.383	.625	.242
15	.387	.602	.214
16	.248	.529	.281
17	.273	.403	.13
18	.243	.428	.185
19	.364	.570	.206
20	.296	.464	.168
21	.340	.614	.275
22	.324	.452	.129
23	.308	.568	.26
24	.231	.465	.234
25	.262	.520	.258
26	.251	.506	.255
27	.234	.355	.121
28	.294	.555	.26
29	.385	.618	.233
30	.267	.371	.104

Note. 1. IRP indicates the item response probability, also known as correct response rate.

Appendix F

True Values of λs^I under the LCDM for Item Pool 2

Item	λ_0 (Intercepts)	λ_1	λ_2	λ_3	λ_{12}	λ_{13}	λ_{23}	λ_{123}
1	-.827	.836	0	0	0	0	0	0
2	-.557	0	.963	0	0	0	0	0
3	-.422	0	0	.534	0	0	0	0
4	-1.034	.566	0	0	0	0	0	0
5	-1.373	0	.313	0	0	0	0	0
6	-.539	0	0	.289	0	0	0	0
7	-1.110	.125	0	.562	0	.267	0	0
8	-.957	0	.130	.336	0	0	.127	0
9	-.901	.200	.391	0	.158	0	0	0
10	-.989	0	.442	.228	0	0	.047	0
11	-.726	.3	.489	0	.183	0	0	0
12	-.972	.480	0	.486	0	-.161	0	0
13	-.656	.240	.275	0	.341	0	0	0
14	-.478	0	.361	.400	0	0	.228	0
15	-.458	.269	0	.249	0	.351	0	0
16	-1.111	.526	.37	0	.331	0	0	0
17	-.980	0	.167	.233	0	0	.186	0
18	-1.135	.265	0	.389	0	.192	0	0
19	-.557	.565	.179	.451	-.154	-.591	-.596	.984
20	-.869	.723	0	0	0	0	0	0
21	-.665	0	1.131	0	0	0	0	0
22	-.737	0	0	.545	0	0	0	0
23	-.808	0	.109	.201	0	0	.771	0
24	-1.205	.504	0	.318	0	.242	0	0
25	-1.034	.583	.475	0	.056	0	0	0
26	-1.091	.300	0	.295	0	.520	0	0
27	-1.188	0	0	.591	0	0	0	0
28	-.875	0	1.094	0	0	0	0	0
29	-.468	.558	.365	0	.027	0	0	0
30	-1.009	.483	0	0	0	0	0	0

Note. 1. λ_0 is intercept; λ_1, λ_2, and λ_3 are main effects; λ_{12}, λ_{13}, λ_{23}, and λ_{123} are interaction effects.

Appendix G

Q-matrix for 3 Attribute, 20 Items Test.

Item	Attribute 1	Attribute 2	Attribute 3
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
5	0	1	0
6	0	0	1
7	0	1	1
8	1	1	0
9	0	0	1
10	1	1	0
11	1	1	1
12	1	0	0
13	1	1	0
14	1	0	1
15	1	1	0
16	0	0	1
17	0	1	0
18	1	0	0
19	1	0	1
20	1	0	1

Appendix H

Q-matrix for 4 Attribute, 20 Items Test.

Item	Attribute	Attribute 2	Attribute 3	Attribute 4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0
6	0	1	0	0
7	0	0	1	0
8	0	0	0	1
9	0	1	1	1
10	1	1	0	1
11	0	0	1	0
12	1	1	0	0
13	1	1	1	1
14	1	0	0	1
15	1	1	0	1
16	1	0	1	0
17	1	1	0	0
18	0	0	1	1
19	0	0	0	1
20	0	1	0	1

Appendix I

Q-matrix for 4 Attribute, 30 Items Test.

Item	Attribute	Attribute 2	Attribute 3	Attribute 4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0
6	0	1	0	0
7	0	0	1	0
8	0	0	0	1
9	0	1	1	1
10	1	1	0	1
11	0	0	1	0
12	1	1	0	0
13	1	1	1	1
14	1	0	0	1
15	1	1	0	1
16	1	0	1	0
17	1	1	0	0
18	0	0	1	1
19	0	0	0	1
20	0	1	0	1
21	1	0	0	0
22	1	0	1	1
23	1	0	1	1
24	0	1	0	1
25	0	0	1	0
26	1	1	1	1
27	0	0	0	1
28	0	1	1	1
29	0	1	0	1
30	1	0	0	1

Appendix J

IRP* Table of 3 Attribute, 20 Items, High Discrimination Test.

Item	Non-Mastery Group IRP	Mastery Group IRP	Discrimination
1	.324	.699	.375
2	.342	.757	.415
3	.172	.878	.706
4	.357	.821	.465
5	.234	.771	.537
6	.246	.887	.641
7	.151	.698	.547
8	.334	.816	.482
9	.295	.823	.527
10	.3	.898	.598
11	.197	.729	.533
12	.25	.697	.447
13	.261	.703	.441
14	.212	.853	.641
15	.218	.877	.66
16	.145	.828	.682
17	.218	.754	.536
18	.256	.855	.598
19	.289	.822	.533
20	.27	.751	.481

Note. * IRP indicates the item response probability, also known as correct response rate.

Appendix K

IRP* Table of 3 Attribute, 20 Items, Mixed Discrimination Test.

Item	Non-Mastery Group IRP	Mastery Group IRP	Discrimination
1	.354	.586	.232
2	.364	.814	.45
3	.366	.507	.141
4	.288	.776	.488
5	.167	.731	.563
6	.39	.63	.24
7	.208	.795	.586
8	.213	.758	.545
9	.205	.678	.473
10	.248	.467	.218
11	.225	.856	.631
12	.331	.35	.019
13	.296	.87	.574
14	.159	.83	.671
15	.306	.412	.106
16	.308	.694	.386
17	.273	.463	.19
18	.224	.342	.119
19	.417	.674	.257
20	.202	.856	.654

Note. * IRP indicates the item response probability, also known as correct response rate.

Appendix L

IRP* Table of 4 Attribute, 20 Items, High Discrimination Test.

Item	Non-Mastery Group IRP	Mastery Group IRP	Discrimination
1	.33	.711	.381
2	.35	.778	.428
3	.159	.892	.733
4	.336	.75	.415
5	.252	.796	.544
6	.239	.896	.657
7	.274	.819	.545
8	.328	.705	.378
9	.205	.697	.492
10	.238	.762	.524
11	.192	.692	.501
12	.195	.803	.608
13	.302	.818	.516
14	.261	.726	.465
15	.184	.739	.555
16	.356	.757	.402
17	.172	.829	.657
18	.319	.754	.436
19	.175	.683	.508
20	.303	.839	.537

Note. * IRP indicates the item response probability, also known as correct response rate.

Appendix M

IRP* Table of 4 Attribute, 20 Items, Mixed Discrimination Test.

Item	Non-Mastery Group IRP	Mastery Group IRP	Discrimination
1	.394	.57	.176
2	.35	.794	.444
3	.335	.526	.19
4	.254	.762	.508
5	.208	.715	.507
6	.396	.662	.266
7	.331	.823	.492
8	.278	.642	.363
9	.388	.73	.342
10	.186	.422	.236
11	.253	.881	.627
12	.436	.522	.086
13	.265	.723	.458
14	.198	.777	.579
15	.273	.406	.134
16	.372	.753	.381
17	.23	.475	.244
18	.351	.55	.199
19	.242	.262	.02
20	.189	.794	.604

Note. * IRP indicates the item response probability, also known as correct response rate.

Appendix N

IRP* Table of 4 Attribute, 30 Items, High Discrimination Test.

Item	Non-Mastery Group IRP	Mastery Group IRP	Discrimination
1	.307	.736	.429
2	.349	.774	.425
3	.141	.898	.757
4	.333	.745	.412
5	.248	.782	.535
6	.248	.878	.63
7	.256	.846	.59
8	.324	.68	.356
9	.199	.738	.539
10	.236	.78	.544
11	.175	.702	.527
12	.188	.791	.603
13	.251	.869	.617
14	.283	.722	.439
15	.215	.694	.479
16	.343	.739	.396
17	.171	.836	.665
18	.376	.742	.365
19	.167	.685	.518
20	.282	.842	.56
21	.229	.898	.668
22	.211	.753	.542
23	.189	.872	.683
24	.261	.717	.457
25	.263	.867	.603
26	.219	.884	.665
27	.297	.744	.447
28	.234	.792	.558
29	.331	.734	.403
30	.245	.795	.55

Note. * IRP indicates the item response probability, also known as correct response rate.

Appendix O

IRP* Table of 4 Attribute, 30 Items, Mixed Discrimination Test.

Item	Non-Mastery Group IRP	Mastery Group IRP	Discrimination
1	.405	.681	.275
2	.325	.828	.504
3	.197	.366	.169
4	.195	.856	.661
5	.21	.69	.481
6	.248	.347	.099
7	.214	.756	.541
8	.195	.683	.488
9	.199	.898	.699
10	.379	.455	.076
11	.195	.786	.592
12	.264	.539	.274
13	.179	.805	.627
14	.202	.89	.688
15	.2	.502	.302
16	.265	.787	.522
17	.347	.593	.246
18	.26	.408	.148
19	.307	.364	.057
20	.193	.827	.634
21	.216	.76	.544
22	.216	.88	.664
23	.271	.784	.513
24	.224	.782	.558
25	.173	.727	.553
26	.296	.677	.381
27	.276	.768	.491
28	.264	.833	.568
29	.255	.403	.148
30	.334	.532	.198

Note. * IRP indicates the item response probability, also known as correct response rate.

Appendix P

Π of 3 Attribute, 20 Items, High Discrimination Test.

Item	C1	C2	C3	C4	C5	C6	C7	C8
	000	100	010	110	001	101	011	111
1	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***
2	.327*	.327*	.752***	.752***	.327*	.327*	.752***	.752***
3	.159*	.159*	.159*	.159*	.885***	.885***	.885***	.885***
4	.328*	.782***	.328*	.782***	.328*	.782***	.328*	.782***
5	.241*	.241*	.788***	.788***	.241*	.241*	.788***	.788***
6	.241*	.241*	.241*	.241*	.889***	.889***	.889***	.889***
7	.158*	.158*	.536**	.515**	.421**	.58**	.712***	.712***
8	.349*	.466**	.591**	.81***	.349*	.578**	.539**	.81***
9	.292*	.292*	.292*	.292*	.814***	.814***	.814***	.814***
10	.33*	.509**	.519**	.891***	.33*	.458**	.429**	.891***
11	.196*	.538**	.559**	.405**	.496**	.552**	.443**	.73***
12	.233*	.686***	.233*	.686***	.233*	.686***	.233*	.686***
13	.243*	.483**	.474**	.708***	.243*	.43**	.428**	.708***
14	.174*	.453**	.174*	.572**	.409**	.85***	.488**	.85***
15	.225*	.512**	.441**	.874***	.225*	.426**	.551**	.874***
16	.169*	.169*	.169*	.169*	.816***	.816***	.816***	.816***
17	.205*	.205*	.746***	.746***	.205*	.205*	.746***	.746***
18	.24*	.854***	.24*	.854***	.24*	.854***	.24*	.854***
19	.276*	.562**	.276*	.562**	.559**	.839***	.488**	.839***
20	.273*	.542**	.273*	.4**	.495**	.745***	.444**	.745***

Note. * indicates the π_{ic} for non mastery group, ** indicates the π_{ic} for partial mastery group, *** indicates the π_{ic} for mastery group. The binary vector (e.g., 001) under class name (e.g., C5) indicates the attribute profile of the latent class.

Appendix Q

Π of 3 Attribute, 20 Items, Mixed Discrimination Test.

Item	C1 000	C2 100	C3 010	C4 110	C5 001	C6 101	C7 011	C8 111
1	.378*	.579***	.378*	.579***	.378*	.579***	.378*	.579***
2	.349*	.349*	.81***	.81***	.349*	.349*	.81***	.81***
3	.331*	.331*	.331*	.331*	.537***	.537***	.537***	.537***
4	.269*	.786***	.269*	.786***	.269*	.786***	.269*	.786***
5	.179*	.179*	.722***	.722***	.179*	.179*	.722***	.722***
6	.393*	.393*	.393*	.393*	.654***	.654***	.654***	.654***
7	.193*	.193*	.538**	.559**	.405**	.496**	.84***	.84***
8	.224*	.464**	.446**	.753***	.224*	.429**	.483**	.753***
9	.178*	.178*	.178*	.178*	.688***	.688***	.688***	.688***
10	.247*	.317**	.287**	.447***	.247*	.375**	.253**	.447***
11	.225*	.56**	.424**	.512**	.441**	.426**	.551**	.874***
12	.333*	.361***	.333*	.361***	.333*	.361***	.333*	.361***
13	.312*	.477**	.455**	.853***	.312*	.563**	.49**	.853***
14	.15*	.559**	.15*	.488**	.551**	.828***	.526**	.828***
15	.295*	.328**	.352**	.409***	.295*	.387**	.348**	.409***
16	.284*	.284*	.284*	.284*	.711***	.711***	.711***	.711***
17	.284*	.284*	.512***	.512***	.284*	.284*	.512***	.512***
18	.221*	.347***	.221*	.347***	.221*	.347***	.221*	.347***
19	.397*	.531**	.397*	.53**	.423**	.633***	.417**	.633***
20	.169*	.469**	.169*	.531**	.464**	.846***	.438**	.846***

Note. * indicates the π_{ic} for non mastery group, ** indicates the π_{ic} for partial mastery group, *** indicates the π_{ic} for mastery group. The binary vector (e.g., 001) under class name (e.g., C5) indicates the attribute profile of the latent class.

Appendix R

Π of 4 Attribute, 20 Items, High Discrimination Test.

Item	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
	0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111
1	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***
2	.327*	.327*	.752***	.752***	.327*	.327*	.752***	.752***	.327*	.327*	.752***	.752***	.327*	.327*	.752***	.752***
3	.159*	.159*	.159*	.159*	.885***	.885***	.885***	.885***	.159*	.159*	.159*	.159*	.885***	.885***	.885***	.885***
4	.328*	.328*	.328*	.328*	.328*	.328*	.328*	.328*	.782***	.782***	.782***	.782***	.782***	.782***	.782***	.782***
5	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***
6	.241*	.241*	.889***	.889***	.241*	.241*	.889***	.889***	.241*	.241*	.889***	.889***	.241*	.241*	.889***	.889***
7	.265*	.265*	.265*	.265*	.819***	.819***	.819***	.819***	.265*	.265*	.265*	.265*	.819***	.819***	.819***	.819***
8	.33*	.33*	.33*	.33*	.33*	.33*	.33*	.33*	.676***	.676***	.676***	.676***	.676***	.676***	.676***	.676***
9	.179*	.179*	.449**	.408**	.466**	.591**	.578**	.539**	.528**	.599**	.531**	.542**	.509**	.519**	.722***	.722***
10	.224*	.593**	.58**	.538**	.224*	.559**	.405**	.496**	.552**	.443**	.464**	.753***	.446**	.429**	.483**	.753***
11	.178*	.178*	.178*	.178*	.688***	.688***	.688***	.688***	.178*	.178*	.178*	.178*	.688***	.688***	.688***	.688***
12	.191*	.447**	.493**	.79***	.191*	.453**	.572**	.79***	.191*	.409**	.488**	.79***	.191*	.56**	.424**	.79***
13	.276*	.426**	.551**	.579**	.475**	.533**	.419**	.477**	.455**	.563**	.49**	.562**	.562**	.559**	.488**	.839***
14	.284*	.542**	.284*	.4**	.284*	.495**	.284*	.444**	.476**	.711***	.523**	.711***	.47**	.711***	.422**	.711***
15	.188*	.484**	.558**	.421**	.188*	.487**	.597**	.579**	.577**	.435**	.426**	.73***	.531**	.469**	.531**	.73***
16	.328*	.556**	.328*	.419**	.493**	.771***	.502**	.771***	.328*	.52**	.328*	.467**	.498**	.771***	.591**	.771***
17	.178*	.583**	.522**	.83***	.178*	.482**	.429**	.83***	.178*	.587**	.46**	.83***	.178*	.412**	.59**	.83***
18	.347*	.347*	.347*	.347*	.51**	.591**	.517**	.481**	.53**	.464**	.462**	.444**	.742***	.742***	.742***	.742***
19	.168*	.168*	.168*	.168*	.168*	.168*	.168*	.168*	.689***	.689***	.689***	.689***	.689***	.689***	.689***	.689***
20	.307*	.307*	.428**	.538**	.307*	.307*	.524**	.578**	.535**	.547**	.855***	.855***	.504**	.532**	.855***	.855***

Note. * indicates the π_{ic} for non mastery group, ** indicates the π_{ic} for partial mastery group, *** indicates the π_{ic} for mastery group. The binary vector (e.g., 0010) under class name (e.g., C5) indicates the attribute profile of the latent class.

Appendix S

Π of 4 Attribute, 20 Items, Mixed Discrimination Test.

Item	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
	0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111
1	.378*	.579***	.378*	.579***	.378*	.579***	.378*	.579***	.378*	.579***	.378*	.579***	.378*	.579***	.378*	.579***
2	.349*	.349*	.81***	.81***	.349*	.349*	.81***	.81***	.349*	.349*	.81***	.81***	.349	.349	.81***	.81***
3	.331*	.331*	.331*	.331*	.537***	.537***	.537***	.537***	.331*	.331*	.331*	.331*	.537***	.537***	.537***	.537***
4	.269*	.269*	.269*	.269*	.269*	.269*	.269*	.269*	.786***	.786***	.786***	.786***	.786***	.786***	.786***	.786***
5	.179*	.722***	.179*	.722***	.179*	.722***	.179*	.722***	.179*	.722***	.179*	.722***	.179*	.722***	.179*	.722***
6	.393*	.393*	.654***	.654***	.393*	.393*	.654***	.654***	.393*	.393*	.654***	.654***	.393*	.393*	.654***	.654***
7	.309*	.309*	.309*	.309*	.823***	.823***	.823***	.823***	.309*	.309*	.309*	.309*	.823***	.823***	.823***	.823***
8	.246*	.246*	.246*	.246*	.246*	.246*	.246*	.246	.656***	.656***	.656***	.656***	.656***	.656***	.656***	.656***
9	.322*	.322*	.552**	.443**	.464**	.446**	.429**	.483**	.483**	.474**	.43**	.428**	.447***	.493	.716***	.716***
10	.209*	.275**	.329**	.227**	.209*	.293**	.24**	.228**	.322**	.343***	.265**	.386***	.309**	.223**	.267**	.386***
11	.24*	.24*	.24*	.24*	.854***	.854***	.854***	.854***	.24*	.24*	.24*	.24*	.854***	.854***	.854***	.854***
12	.362*	.484**	.481**	.521***	.362*	.428**	.475**	.521***	.362*	.456**	.469**	.521***	.362*	.362*	.433**	.521***
13	.281*	.476**	.523**	.47**	.422**	.449**	.534**	.484**	.558**	.421**	.487**	.597**	.579**	.577**	.435**	.683***
14	.217*	.469**	.217*	.531**	.217*	.464**	.217*	.438**	.556**	.8***	.419**	.8***	.493**	.8***	.502**	.8***
15	.298*	.441**	.37**	.431**	.298*	.435**	.389**	.359**	.32**	.438**	.343	.462***	.307**	.44**	.406**	.462***
16	.347*	.51**	.347*	.591**	.517**	.742***	.481**	.742***	.347*	.53**	.347*	.464	.462**	.742***	.444**	.742***
17	.231*	.244**	.252**	.468***	.231*	.334**	.324**	.468***	.231*	.365**	.332	.468***	.231*	.341**	.309**	.468***
18	.364*	.364*	.364*	.364*	.482**	.511**	.43**	.411**	.426**	.366**	.392	.491	.544***	.544***	.544***	.544***
19	.248*	.248*	.248*	.248*	.248*	.248*	.248*	.248*	.27***	.27***	.27***	.27***	.27***	.27***	.27***	.27***
20	.193*	.193*	.449**	.546**	.193*	.193*	.569**	.5**	.478**	.449**	.793***	.793***	.422**	.478**	.793***	.793***

Note. * indicates the π_{ic} for non mastery group, ** indicates the π_{ic} for partial mastery group, *** indicates the π_{ic} for mastery group. The binary vector (e.g., 0010) under class name (e.g., C5) indicates the attribute profile of the latent class.

Appendix T

Π of 4 Attribute, 30 Items, High Discrimination Test.

Item	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
	0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111
1	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***	.308*	.722***
2	.327*	.327*	.752***	.752***	.327*	.327*	.752***	.752***	.327*	.327*	.752***	.752***	.327*	.327*	.752***	.752***
3	.159*	.159*	.159*	.159*	.885***	.885***	.885***	.885***	.159*	.159*	.159*	.159*	.885***	.885***	.885***	.885***
4	.328*	.328*	.328*	.328*	.328*	.328*	.328*	.328*	.782***	.782***	.782***	.782***	.782***	.782***	.782***	.782***
5	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***	.241*	.788***
6	.241*	.241*	.889***	.889***	.241*	.241*	.889***	.889***	.241*	.241*	.889***	.889***	.241*	.241*	.889***	.889***
7	.265*	.265*	.265*	.265*	.819***	.819***	.819***	.819***	.265*	.265*	.265*	.265*	.819***	.819***	.819***	.819***
8	.33*	.33*	.33*	.33*	.33*	.33*	.33*	.33*	.676***	.676***	.676***	.676***	.676***	.676***	.676***	.676***
9	.179*	.179*	.449**	.408**	.466**	.591**	.578**	.539**	.528**	.599**	.531**	.542**	.509**	.519**	.722***	.722***
10	.224*	.593**	.58**	.538**	.224*	.559**	.405**	.496**	.552**	.443**	.464**	.753***	.446**	.429**	.483**	.753***
11	.178*	.178*	.178*	.178*	.688***	.688***	.688***	.688***	.178*	.178*	.178*	.178*	.688***	.688***	.688***	.688***
12	.191*	.447**	.493**	.79***	.191*	.453**	.572**	.79***	.191*	.409**	.488**	.79***	.191*	.56**	.424**	.79***
13	.276*	.426**	.551**	.579**	.475**	.533**	.419**	.477**	.455**	.563**	.49**	.562**	.562**	.559**	.488**	.839***
14	.284*	.542**	.284*	.4**	.284*	.495**	.284*	.444**	.476**	.711***	.523**	.711***	.47**	.711***	.422**	.711***
15	.188*	.484**	.558**	.421**	.188*	.487**	.597**	.579**	.577**	.435**	.426**	.73***	.531**	.469**	.531**	.73***
16	.328*	.556**	.328*	.419**	.493**	.771***	.502**	.771***	.328*	.52**	.328*	.467**	.498**	.771***	.591**	.771***
17	.178*	.583**	.522**	.83***	.178*	.482**	.429**	.83***	.178*	.587**	.46**	.83***	.178*	.412**	.59**	.83***
18	.347*	.347*	.347*	.347*	.51**	.591**	.517**	.481**	.53**	.464**	.462**	.444**	.742***	.742***	.742***	.742***
19	.168*	.168*	.168*	.168*	.168*	.168*	.168*	.168*	.689***	.689***	.689***	.689***	.689***	.689***	.689***	.689***
20	.307*	.307*	.428**	.538**	.307*	.307*	.524**	.578**	.535**	.547**	.855***	.855***	.504**	.532**	.855***	.855***
21	.238*	.895***	.238*	.895***	.238*	.895***	.238*	.895***	.238*	.895***	.238*	.895***	.238*	.895***	.238*	.895***
22	.199*	.462**	.199*	.482**	.402**	.437**	.569**	.446**	.448**	.415**	.449**	.546**	.569**	.747***	.5**	.747***
23	.194*	.422**	.194*	.478**	.514**	.443**	.489**	.444**	.5**	.471**	.53**	.475**	.471**	.835***	.507**	.835***
24	.256*	.256*	.483**	.453**	.256*	.256*	.526**	.437**	.573**	.549**	.743***	.743***	.534**	.524**	.743***	.743***
25	.266*	.266*	.266*	.266*	.869***	.869***	.869***	.869***	.266*	.266*	.266*	.266*	.869***	.869***	.869***	.869***
26	.243*	.568**	.462**	.542**	.453**	.519**	.496**	.453**	.513**	.583**	.58**	.455**	.464**	.597**	.524**	.884***
27	.282*	.282*	.282*	.282*	.282*	.282*	.282*	.282*	.752***	.752***	.752***	.752***	.752***	.752***	.752***	.752***
28	.201*	.201*	.43**	.515**	.448**	.592**	.52**	.503**	.481**	.576**	.473**	.458**	.434**	.434**	.771***	.771***
29	.342*	.342*	.443**	.535**	.342*	.342*	.41**	.54**	.47**	.482**	.721***	.721***	.564**	.584**	.721***	.721***
30	.236*	.546**	.236*	.537**	.236*	.411**	.236*	.479**	.496**	.805***	.512**	.805***	.54**	.805***	.583**	.805***

Note. * indicates the π_{ic} for non mastery group, ** indicates the π_{ic} for partial mastery group, *** indicates the π_{ic} for mastery group. The binary vector (e.g., 0010) under class name (e.g., C5) indicates the attribute profile of the latent class.

Appendix U

Π of 4 Attribute, 30 Items, Mixed Discrimination Test.

Item	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
	0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111
1	.393*	.654***	.393*	.654***	.393*	.654***	.393*	.654***	.393*	.654***	.393*	.654***	.393*	.654***	.393*	.654***
2	.309*	.309*	.823***	.823***	.309*	.309*	.823***	.823***	.309*	.309*	.823***	.823***	.309*	.309*	.823***	.823***
3	.205*	.205*	.205*	.205*	.343***	.343***	.343***	.343***	.205*	.205*	.205*	.205*	.343***	.343***	.343***	.343***
4	.193*	.193*	.193*	.193*	.193*	.193*	.193*	.193*	.84***	.84***	.84***	.84***	.84***	.84***	.84***	.84***
5	.196*	.73***	.196*	.73***	.196*	.73***	.196*	.73***	.196*	.73***	.196*	.73***	.196*	.73***	.196*	.73***
6	.229*	.229*	.349***	.349***	.229*	.229*	.349***	.349***	.229*	.229*	.349***	.349***	.229*	.229*	.349***	.349***
7	.224*	.224*	.224*	.224*	.753***	.753***	.753***	.753***	.224*	.224*	.224*	.224*	.753***	.753***	.753***	.753***
8	.178*	.178*	.178*	.178*	.178*	.178*	.178*	.178*	.688***	.688***	.688***	.688***	.688***	.688***	.688***	.688***
9	.225*	.225*	.447**	.493**	.453**	.572**	.409**	.488**	.56**	.424**	.512**	.441**	.426**	.551**	.874***	.874***
10	.333*	.347**	.391**	.374**	.333*	.455***	.4**	.445**	.455***	.452**	.399**	.455***	.446**	.427**	.44**	.455***
11	.194*	.194*	.194*	.194*	.769***	.769***	.769***	.769***	.194*	.194*	.194*	.194*	.769***	.769***	.769***	.769***
12	.276*	.368**	.329**	.469***	.276*	.293**	.313**	.469***	.276*	.376**	.339**	.469***	.276*	.394**	.291**	.469***
13	.217*	.597**	.579**	.577**	.435**	.426**	.531**	.469**	.531**	.464**	.438**	.556**	.419**	.493**	.502**	.8***
14	.21*	.498**	.21*	.591**	.21*	.497**	.21*	.578**	.583**	.884***	.522**	.884***	.482**	.884***	.429**	.884***
15	.212*	.354**	.32**	.233**	.212*	.295**	.355**	.3**	.273**	.309**	.26**	.5***	.258**	.245**	.268**	.5***
16	.282*	.431**	.282*	.418**	.428**	.78***	.538**	.78***	.282*	.524**	.282*	.578**	.535**	.78***	.547**	.78***
17	.364*	.482**	.511**	.544***	.364*	.43**	.411**	.544***	.364*	.426**	.366**	.544***	.364*	.392**	.491**	.544***
18	.248*	.248*	.248*	.248*	.259**	.285**	.358**	.375**	.322**	.306**	.285**	.264**	.438***	.438***	.438***	.438***
19	.314*	.314*	.314*	.314*	.314*	.314*	.314*	.314*	.377***	.377***	.377***	.377***	.377***	.377***	.377***	.377***
20	.194*	.194*	.489**	.444**	.194*	.194*	.5**	.471**	.53**	.475**	.835***	.835***	.471**	.507**	.835***	.835***
21	.203*	.753***	.203*	.753***	.203*	.753***	.203*	.753***	.203*	.753***	.203*	.753***	.203*	.753***	.203*	.753***
22	.203*	.526**	.203*	.437**	.573**	.549**	.534**	.524**	.474**	.506**	.575**	.516**	.568**	.827***	.462**	.827***
23	.282*	.519**	.282*	.496**	.453**	.513**	.583**	.58**	.455**	.464**	.597**	.524**	.587**	.752***	.493**	.752***
24	.208*	.208*	.43**	.515**	.208*	.208*	.448**	.592**	.52**	.503**	.741***	.741***	.481**	.576**	.741***	.741***
25	.184*	.184*	.184*	.184*	.693***	.693***	.693***	.693***	.184*	.184*	.184*	.184*	.693***	.693***	.693***	.693***
26	.229*	.496**	.451**	.443**	.535**	.41**	.54**	.47**	.482**	.564**	.584**	.457**	.592**	.546**	.537**	.663***
27	.262*	.262*	.262*	.262*	.262*	.262*	.262*	.262*	.769***	.769***	.769***	.769***	.769***	.769***	.769***	.769***
28	.282*	.282*	.54**	.583**	.524**	.486**	.508**	.412**	.452**	.479**	.44**	.566**	.431**	.561**	.787***	.787***
29	.234*	.234*	.329**	.281**	.234*	.234*	.343**	.294**	.38**	.379**	.412***	.412***	.343**	.273**	.412***	.412***
30	.319*	.359**	.319*	.398**	.319*	.436**	.319*	.344**	.379**	.591***	.389**	.591***	.449**	.591***	.457**	.591***

Note. * indicates the π_{ic} for non mastery group, ** indicates the π_{ic} for partial mastery group, *** indicates the π_{ic} for mastery group. The binary vector (e.g., 0010) under class name (e.g., C5) indicates the attribute profile of the latent class.

Appendix V

R Code of Data Simulation

```
library(readr)
library(dplyr)
library(purrr)
library(tidyr)
library(ggplot2)
library(glmnet)
library(prodlim)
library(compositions)

source("simFun.R")

### list based function method
##########################
# q-matrix and attribute
# profile loading and parameter initialization

file_path <- "4_30/"
q.mat <-
    read_csv(paste(file_path, "Qmatrix.csv", sep = ''))
q.mat.full <-
    read_csv(paste(file_path, "full_Qmatrix.csv", sep = ''))

#q.mat <- q.mat[1:30, ]
#q.mat.full <- q.mat.full[1:30, ]
SET_DOUBLE_DISTRIBUTION <- FALSE
HIGH <- FALSE
```

```
set.seed(123)

if(SET_DOUBLE_DISTRIBUTION){
  # rep 3 times for two types of discrimination: high low
  q.mat <- bind_rows(q.mat, q.mat)
  q.mat.full <- bind_rows(q.mat.full, q.mat.full)
  n.item <- nrow(q.mat)
  n.attr <- ncol(q.mat)

  dis.idx <- c(rep(TRUE, n.item/2), rep(FALSE, n.item/2))
  #write_csv(data.frame(dis.idx), paste(file_path,
  'dis_indx.csv', sep = '') )
}else{
  n.item <- nrow(q.mat)
  n.attr <- ncol(q.mat)

  if(HIGH) {
    dis.idx <- rep(TRUE, n.item)
  } else {
    dis.idx <- sample(c(TRUE, FALSE),
                      size = n.item, replace = TRUE,
                      prob = c(0.5, 0.5))
  }

  #write_csv(data.frame(dis.idx), paste(file_path,
  'dis_indx.csv', sep = '') )
}

# dis.idx <- c(rep(TRUE, 30), rep(FALSE, 30))
n.class <- 2^n.attr
n.examinee <- 1000 # number of exmainees
cor.attr <- 0.5 # correlation between attributes


attr <- map(1:n.attr, function(x){ 0:1 })
names(attr) <- paste("attr", 1:n.attr, sep = "")
attr.prof <- matrix(unlist(expand.grid(attr)),
ncol = n.attr)
colnames(attr.prof) <- names(attr)

attr.prof.full <- t(apply(attr.prof, 1, comb1, n = 3))
#n = min(c(3, n.attr))))
```

```r
# prob.ini
# first two columns
prob.ini <- c(0.9, 0.65, 0.70, 0.55,
              0.60, 0.40, 0.55, 0.45,
              0.35, 0.15, 0.4, 0.2) %>%
  matrix(ncol = 4, byrow = TRUE)
rownames(prob.ini) <- c("Mas", "parMas", "noMas")
colnames(prob.ini) <- c("upper", "lower",
"upper-low", "lower-low")
# normal discrimination, low discrimination



mat.prob <- attr.prof.full %*% t(as.matrix(q.mat.full))
ideal.prob <- apply(q.mat.full, 1, sum)
lambda <- {}


### set
for(i in 1:n.item){
  if(dis.idx[i]){
    mat.prob[!(mat.prob[, i] %in% c(0, ideal.prob[i])), i] <-
      runif(sum(!(mat.prob[, i] %in% c(0, ideal.prob[i]))),
            min = prob.ini['parMas', 'lower'],
            max = prob.ini['parMas', 'upper']) %>%
      round(3)

    mat.prob[mat.prob[, i] == ideal.prob[i], i] <-
      runif(sum(mat.prob[, i] == ideal.prob[i]),
            min = prob.ini['Mas', 'lower'],
            max = prob.ini['Mas', 'upper']) %>%
      round(4)
    mat.prob[mat.prob[, i] == 0, i] <-
      runif(sum(mat.prob[, i] == 0),
            min = prob.ini['noMas', 'lower'],
            max = prob.ini['noMas', 'upper']) %>%
      round(5)
  }else{
    prob.non <- runif(sum(mat.prob[, i] == 0),
                      min = prob.ini['noMas', 'lower-low'],
                      max = prob.ini['noMas', 'upper-low'])
```

```
    prob.part <-
    runif(sum(!(mat.prob[, i] %in% c(0, ideal.prob[i]))),
    min =  max(prob.non), max =  max(prob.non) + 0.15)
    if(length(prob.part)>0){
      prob.mastery <-
      runif(sum(mat.prob[, i] == ideal.prob[i]),
      min = max(prob.part), max = max(prob.non) + 0.30)
    } else {
      prob.mastery <-
      runif(sum(mat.prob[, i] == ideal.prob[i]),
      min = max(prob.non), max = max(prob.non) + 0.30)
    }

    mat.prob[!(mat.prob[,i] %in% c(0,ideal.prob[i])),i] <-
    prob.part
    mat.prob[mat.prob[, i] == 0, i] <-prob.non
    mat.prob[mat.prob[, i] == ideal.prob[i], i] <-
    prob.mastery
  }

  mat.k <- cbind(1,
  attr.prof.full*
  matrix(rep(unlist(q.mat.full[i, ]), n.class),
  nrow = n.class, byrow = TRUE))

  prob <- mat.prob[, i]
  lambda <- cbind(lambda,
  round(MASS::ginv(mat.k) %*% log(prob/(1-prob)), 4)
  )
}

pi <- t(mat.prob) %>%
  data.frame()
names(pi) <- 1:ncol(pi)

if(HIGH) {
  write_csv(round(pi, 3),
  file.path(file_path, 'pi_high.csv'))
} else {
  write_csv(round(pi, 3),
  file.path(file_path, 'pi_low.csv'))
```

```
}


### set different prob for different attribute

lambda <- data.frame(t(lambda))
colnames(lambda) <- c('Intercept', colnames(q.mat.full))

## save the item parameters
if(HIGH) {
  write_csv(lambda,
    paste(file_path, 'lambda high.csv', sep = ''))
} else {
  write_csv(lambda,
    paste(file_path, 'lambda mixed.csv', sep = ''))
}



# ################################################
# Simulate Attribute profile and get the response
# ################################################

avg.attr <- rep(0.5, n.attr)

sigma <- matrix(cor.attr, n.attr, n.attr)
diag(sigma) <- 1
examinee.attr <- bindata::rmvbin(n.examinee,
                                 margprob = avg.attr,
                                 sigma = sigma)

examinee.category <-
    apply(examinee.attr, 1, prodlim::row.match, attr.prof)
examinee.attr <-
    cbind(examinee.attr, examinee.category)
colnames(examinee.attr) <- c(names(q.mat), 'class')

if(HIGH){
  write_csv(data.frame(examinee.attr),
    paste(file_path, "attributeProfile high.csv", sep = ''))
  write_csv(data.frame(attr.prof),
    paste(file_path, "attributePattern high.csv", sep = ''))
} else {
```

```
  write_csv(data.frame(examinee.attr),
    paste(file_path, "attributeProfile_mixed.csv", sep = ''))
  write_csv(data.frame(attr.prof),
    paste(file_path, "attributePattern_mixed.csv", sep = ''))
}
## get the response per item
## add class for each examinee

response.prob <- mat.prob[examinee.category, ]
response.pattern <-
    (response.prob > matrix(runif(n.item*n.examinee, 0, 1),
nrow = n.examinee))

colnames(response.pattern) <- paste("Item",
sprintf("%02d",1:nrow(q.mat)), sep = "_")


response <- data.frame(response.pattern)
if(HIGH){
  write_csv(response,
    paste(file_path, 'response_high.csv', sep = ''))
} else {
  write_csv(response,
    paste(file_path, 'response_mixed.csv', sep = ''))
}

g1 <- response %>%
  mutate(Class = examinee.category) %>%
  gather(key = Item, value = Response, -Class) %>%
  ggplot(aes(Class, fill = Response)) +
  geom_bar(position = "fill") +
  facet_wrap(~Item, ncol = 10)

if(HIGH){
  ggsave(file.path(file_path, "Item_response_high.pdf"),
         g1, width = 20, height = 40)

} else {
  ggsave(file.path(file_path, "Item_response_mixed.pdf"),
         g1, width = 20, height = 40)
}
```

```
# save IRP
IRP <- NULL
for (i in 1:n.item){
  attr.names <- names(q.mat)[q.mat[i, ] == 1]
  sub.examinee.attr <- examinee.attr[, attr.names] == 1
  if(length(attr.names) > 1){
    mastery.group <- apply(sub.examinee.attr, 1, all)
    nonmastery.group <- apply(!sub.examinee.attr, 1, all)
    partialmastery.group <- !nonmastery.group & !nonmastery.group
    sub.IRP <- c(mean(response[nonmastery.group, i]),
                 mean(response[partialmastery.group, i]),
                 mean(response[mastery.group, i]))
    IRP <- rbind(IRP, sub.IRP)
  } else {
    mastery.group <- sub.examinee.attr
    nonmastery.group <- !sub.examinee.attr
    sub.IRP <- c(mean(response[nonmastery.group, i]),
                 -1,
                 mean(response[mastery.group, i]))
    IRP <- rbind(IRP, sub.IRP)
  }
}

IRP <- data.frame(IRP)
names(IRP) <-
    c("non-mastery group",
    'partially mastery group', 'mastery group')
IRP <- IRP %>%
  mutate(Discrimination =
  `mastery group` - `non-mastery group`)
if(HIGH) {
  write_csv(x = round(IRP, 3),
    path = paste(file_path, 'IRP table high.csv'))
} else {
  write_csv(x = round(IRP, 3),
    path = paste(file_path, 'IRP table mixed.csv'))
}
```

Appendix W

R Code of DCMs fitting

```r
# first test of error labeling method
library(CDM)
library(tidyverse)

## num of attr,
## num of item,
## level of disc,
## q-matrix accuracy,
## examinees

n_attr <- 3
n_item <- 15
disc <- 'h'
acc_q <- 100
n_examinee <- 1000

folder_name <- paste(n_attr,   # num of attr
                     n_item, # num of item
                     disc, # level of disc
                     acc_q, # q-matrix accuracy
                     sep = "_")

save_sub_path <- file.path(save_path, folder_name)
if(!dir.exists(save_sub_path)) dir.create(save_sub_path)

item_list <- c(1:7, 48:52, 73:75)
set.seed(123)
index_examinee <-
    sample(1:2000, 2000, replace = FALSE)[1:1000]
```

```
true_label <-
    read_csv(file.path(data_path,
    "attributeProfile.csv"))[index_examinee, ]
true_attribute_profile <- true_label[,1:n_attr]
true_class <- true_label[ ,n_attr+1]

attr_prof <-
    read_csv(file.path(data_path,
    "attr_prof.csv"))[,-1]
q_mat <-
    read_csv(file.path(data_path,
    "Qmatrix.csv"))[c(1:12,33:35), ]
response <-
    read_csv(file.path(data_path,
    "response.csv"))[index_examinee, item_list]

if(acc_q < 100){
  complex_index = apply(q_mat, 1, sum) > 1
  complex_q_mat = q_mat[complex_index, ] %>%
    as.matrix()
  error_element =
    round((100-acc_q)/100*nrow(q_mat)*ncol(q_mat))
  set.seed(123)
  row_index = sample(x = 1:nrow(complex_q_mat),
                     size = error_element,
                     replace = F)
  col_index = sample(x = 1:ncol(complex_q_mat),
                     size = error_element,
                     replace = T)
  complex_q_mat[cbind(row_index, col_index)] =
    1 - complex_q_mat[cbind(row_index, col_index)]
  q_mat[complex_index, ] = as.matrix(complex_q_mat)
}

# using DINA model to estimate
fit_dina <- gdina(response*1,
    q.matrix = q_mat, rule = "DINA", seed = 123)
post_attr_dina <- fit_dina$pattern %>%
  dplyr::select(starts_with("post.attr"))

attr_dina <-
```

```
    (post_attr_dina > 0.5)*1 == true_attribute_profile

# using DINO model to estimate
fit_dino <-
    gdina(response*1, q.matrix = q_mat,
        rule = "DINO", seed = 123)
post_attr_dino <- fit_dino$pattern %>%
  dplyr::select(starts_with("post.attr"))

attr_dino <-
    (post_attr_dino > 0.5)*1 == true_attribute_profile

# using LCDM model to estimate
fit_lcdm <-
    gdina(response*1, q.matrix = q_mat,
    linkfct = "logit", seed = 123)
post_attr_lcdm <- fit_lcdm$pattern %>%
  dplyr::select(starts_with("post.attr"))

attr_lcdm <-
    (post_attr_lcdm > 0.5)*1 == true_attribute_profile

# using G-DINA model to estimate
fit_gdina <-
    gdina(response*1, q.matrix = q_mat, seed = 123)
post_attr_gdina <- fit_gdina$pattern %>%
  dplyr::select(starts_with("post.attr"))

attr_gdino <-
    (post_attr_gdina > 0.5)*1 == true_attribute_profile

# using RRUM model to estimate
fit_rrum <-
    gdina(response*1, q.matrix = q_mat,
    rule = "RRUM", seed = 123)

post_attr_rrum <- fit_rrum$pattern %>%
  dplyr::select(starts_with("post.attr"))

attr_rrum <-
    (post_attr_rrum > 0.5)*1 == true_attribute_profile
```

```r
# using ACDM model to estimate
fit_acdm <-
    gdina(response*1, q.matrix = q_mat,
    rule = "ACDM", seed = 123)
post_attr_acdm <- fit_acdm$pattern %>%
  dplyr::select(starts_with("post.attr"))

attr_acdm <-
    (post_attr_acdm > 0.5)*1 == true_attribute_profile

#
cat("\n DINA:", apply(attr_dina, 2, mean), "\n",
    "DINO:", apply(attr_dino, 2, mean), "\n",
    "LCDM:", apply(attr_lcdm, 2, mean), "\n",
    "gdina:", apply(attr_gdino, 2, mean), "\n",
    "RRUM:", apply(attr_rrum, 2, mean), "\n",
    "ACDM:", apply(attr_acdm, 2, mean), "\n")


# saving the error labels and true value as csv
write_csv(x = true_label,
    path = file.path(save_sub_path, "true_label.csv"))
write_csv(x = q_mat,
    path = file.path(save_sub_path, "q_mat.csv"))
write_csv(x = response,
    path = file.path(save_sub_path, "response.csv"))

all_attr <-
    cbind(post_attr_dina, post_attr_dino, post_attr_lcdm,
    post_attr_gdina, post_attr_rrum, post_attr_acdm) %>%
  data.frame()
names(all_attr) <-
    paste(paste("attr", c(1:ncol(q_mat)), sep = "_"),
    rep(c("DINA", "DINO", "LCDM", "GDINA", "RRUM", "ACDM"),
    each = ncol(q_mat)), sep = "_")
write_csv(all_attr,
    path = file.path(save_sub_path, "error_label.csv"))
```

Appendix X

Python Code of MAEN in Chapter 3

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 28 11:49:14 2018
This version assume only one hidden layer
between responses (input)
and attribute (code);
@author: Kang Xue
"""

import tensorflow as tf
import numpy as np
import pandas as pd
#import matplotlib.pyplot as plt
import os
from sklearn.cluster import KMeans


dis_index =
    np.array(pd.read_csv('inputData2/dis_indx.csv'))
q_matrix_all =
np.array(pd.read_csv('inputData2/Qmatrix.csv'))
    q_matrix_all = np.row_stack((q_matrix_all, q_matrix_all))
simple_index = np.apply_along_axis(np.sum, 1, q_matrix_all) == 1

item_info = np.column_stack(
    (range(q_matrix_all.shape[0]), simple_index,
```

```
            dis_index, q_matrix_all))
# itemID, simple, discrimination, Q-matrix

# choose the items
loop_inform = pd.read_csv('inputData2/item_selection.csv')

for fileName, item_choice in loop_inform.items():
    print(fileName)
    file_path = 'Results/' + fileName + '/'
    if not os.path.exists(file_path):
        os.mkdir(file_path)

    # load q matrix, response
    q_matrix =
        q_matrix_all[item_choice, :]
    response =
        np.array(pd.read_csv('inputData2/response.csv'))
        [:, item_choice]
    attr_profile =
        np.array(pd.read_csv('inputData2/attributeProfile.csv'))

    n_attr = q_matrix.shape[1] # number of attributes
    n_examinee = response.shape[0] # number of examinees

    sample_index =
        np.random.choice(n_examinee, 2000)
    response = response[sample_index,:]
    simple_index = np.apply_along_axis(np.sum, 1, q_matrix) == 1

    simple_response = response[:, simple_index]
    simple_qmat = q_matrix[simple_index, :]

    complex_response = response[:, ~simple_index]
    complex_qmat = q_matrix[~simple_index, :]

    # Training Parameters
    num_input = np.apply_along_axis(sum, 0, simple_qmat)
    num_input = np.append(num_input, complex_qmat.shape[0])
    cum_input = np.cumsum(num_input)

    num_hidden = 100
    num_hidden_2 = 50
```

```
learning_rate = 0.001
num_steps = 1000
batch_size = 200
loop_max = 50

n_code = n_attr

order_index =
np.concatenate((
np.array(range(q_matrix.shape[0]))
[simple_index & q_matrix[:,0] == 1],
np.array(range(q_matrix.shape[0]))
[simple_index & q_matrix[:,1] == 1],
np.array(range(q_matrix.shape[0]))
[simple_index & q_matrix[:,2] == 1],
np.array(range(q_matrix.shape[0]))
[~simple_index]
),axis = 0)

response = response[:, order_index]

# tf Graph input (only pictures)
X = tf.placeholder("float", [None, cum_input[-1]])

Y = tf.placeholder("float", [None, cum_input[-1]])


weights = {
    'encoder_h1':
    tf.Variable(tf.random_normal(
        [cum_input[-1], num_hidden],
    mean = 0)),
    'encoder_h2':
    tf.Variable(tf.random_normal(
        [num_hidden, n_code], mean = 0)),

    'decoder_h1_1':
    tf.Variable(tf.random_normal(
        [1, num_input[0]], mean = 1)),
    'decoder_h1_2':
    tf.Variable(tf.random_normal(
        [1, num_input[1]], mean = 1)),
```

```
    'decoder_h1_3':
    tf.Variable(tf.random_normal(
        [1, num_input[2]], mean = 1)),
    'decoder_h1_4':
    tf.Variable(tf.random_normal(
        [n_code, num_hidden_2], mean = 1)),
    'decoder_h2':
    tf.Variable(tf.random_normal(
        [num_hidden_2, num_input[3]],
    mean = 1))
    }

biases = {
    'encoder_b1':
    tf.Variable(tf.random_normal(
        [num_hidden], mean = 0)),
    'encoder_b2':
    tf.Variable(tf.random_normal(
        [n_code], mean = 0)),

    'decoder_b1_1':
    tf.Variable(tf.random_normal(
        [num_input[0]], mean = -2.5)),
    'decoder_b1_2':
    tf.Variable(tf.random_normal(
        [num_input[1]], mean = -2.5)),
    'decoder_b1_3':
    tf.Variable(tf.random_normal(
        [num_input[2]], mean = -2.5)),
    'decoder_b1_4':
    tf.Variable(tf.random_normal(
        [num_hidden_2], mean = -2.5)),
    'decoder_b2':
    tf.Variable(tf.random_normal(
        [num_input[3]], mean = -2.5)),

}


def encoder(x):
    # Encoder Hidden layer with sigmoid activation #1
    layer_1 = tf.nn.relu(
```

```
        tf.add(tf.matmul(x, weights['encoder_h1']),
        biases['encoder_b1']))
        layer_2 = tf.nn.sigmoid(
        tf.add(tf.matmul(tf.nn.dropout(layer_1, rate = 0.3),
            weights['encoder_h2']), biases['encoder_b2']))
        return layer_2


# Building the decoder
def decoder(x):
        #x = x > tf.constant(0.5, shape = x.shape)
        # Decoder Hidden layer with sigmoid activation #1
        layer_1_1 = tf.nn.sigmoid(
        tf.add(tf.matmul(x[:,0:1], weights['decoder_h1_1']),
        biases['decoder_b1_1']))
        layer_1_2 = tf.nn.sigmoid(
        tf.add(tf.matmul(x[:,1:2], weights['decoder_h1_2']),
        biases['decoder_b1_2']))
        layer_1_3 = tf.nn.sigmoid(
        tf.add(tf.matmul(x[:,2:3], weights['decoder_h1_3']),
        biases['decoder_b1_3']))

        layer_1_4 = tf.nn.relu(
        tf.add(tf.matmul(x, weights['decoder_h1_4']),
        biases['decoder_b1_4']))

        layer_2 = tf.nn.sigmoid(
        tf.add(tf.matmul(
        tf.nn.dropout(layer_1_4, rate = 0.3),
        weights['decoder_h2']), biases['decoder_b2']))

        layer_3 = tf.concat([layer_1_1, layer_1_2,
        layer_1_3, layer_2], 1)

        return layer_3

encoder_op = encoder(X)

decoder_op = decoder(encoder_op)

# Prediction
y_pred = decoder_op
```

```
# Targets (Labels) are the input data.
y_true = Y

# Define loss and optimizer, minimize the squared error
loss = tf.reduce_mean(tf.pow(y_true - y_pred, 2))
optimizer =
    tf.train.RMSPropOptimizer(learning_rate).minimize(loss)

init = tf.global_variables_initializer()

def q_matrix_detect(attr_est, i):
    noMas = np.array([response[~attr_est[:,0], i].mean(),
                      response[~attr_est[:,1], i].mean(),
                      response[~attr_est[:,2], i].mean()])
    Mas = np.array([response[attr_est[:,0], i].mean(),
                    response[attr_est[:,1], i].mean(),
                    response[attr_est[:,2], i].mean()])

    dif = Mas - noMas

    kmeans = KMeans(n_clusters = 2).fit(dif.reshape(-1,1))

    # label switch
    if(dif[kmeans.labels_ == 0].mean() >
        dif[kmeans.labels_ == 1].mean()):
        kmeans.labels_ = np.abs(kmeans.labels_ - 1)

    if(kmeans.labels_.sum() <= 1):
        kmeans.labels_ = (dif > 0.15)*1
        if(kmeans.labels_.sum() <= 1):
            kmeans.labels_[np.argsort(dif)[[-1, -2]]] = 1

    return kmeans.labels_


pattern_accuracy = []
attribute_accuracy = []
Q_error_rate = []
loop = 0
while(loop < loop_max):
    with tf.Session() as sess:
        # Run the initializer
```

```
    sess.run(init)

    # Training
    for i in range(1, num_steps+1):
        # Prepare Data
        # Get the next batch of response data

        start = (i-1)*batch_size%response.shape[0]
        if(start == 0):
            rand_index =
            np.random.choice(response.shape[0],
                             response.shape[0])

        batch_x =
            response[rand_index[
                start:(start+batch_size)], :]
        batch_y =
            response[rand_index[
                start:(start+batch_size)],:]



        # Run optimization op (backprop)
        # and cost op (to get loss value)
        _, l = sess.run([optimizer, loss],
                feed_dict={X: batch_x, Y: batch_y})
        # Display logs per step
        if i%1000 == 0:
            print('Step %i: Minibatch Loss: %f'
                % (i, l))

    code = sess.run(encoder_op,
                feed_dict={X: response})

attr_est = code > 0.5
failure = any(np.apply_along_axis(np.mean, 0, attr_est) ==
any(np.apply_along_axis(np.mean, 0, attr_est) == 0)
if(failure == False):
# determine the attribute true or false use simple item
    for i in range(attr_est.shape[1]):
        y1 =
            simple_response[attr_est[:,i],:][:,
            simple_qmat[:,i]==1]
```

```
            y2 =
                simple_response[~attr_est[:,i],:][:,
                simple_qmat[:,i]==1]
            if(np.mean(np.apply_along_axis(np.mean, 1, y1)) <
             np.mean(np.apply_along_axis(np.mean, 1, y2))):
                attr_est[:, i] = ~attr_est[:, i]

        d =
         attr_est*1 == attr_profile[sample_index, 0:n_attr]

        accuracy =
            np.mean(np.apply_along_axis(np.prod, 1, d))
        pattern_accuracy =
            np.append(pattern_accuracy, accuracy)
        attribute_accuracy =
            np.append(attribute_accuracy,
                np.apply_along_axis(np.mean, 0, d),
                axis = 0)
        loop += 1
        print(loop)

        q_matrix_est = []
        for i in range(q_matrix.shape[0]):
            if(~simple_index[order_index][i]):
                q_matrix_est =
                    np.append(q_matrix_est,
                        q_matrix_detect(attr_est, i))
            else:
                q_matrix_est =
                    np.append(q_matrix_est,
                        q_matrix[order_index, :][i, :])
    #print(q_matrix_detect(attr_est, i))
        q_matrix_est =
            q_matrix_est.reshape([q_matrix.shape[0], -1])

# print(q_matrix_est)
# print(q_matrix[order_index, :])
        q_diff =
            q_matrix_est - q_matrix[order_index, :]
        q_diff =
            q_diff[np.apply_along_axis(
                sum, 1, q_matrix) > 1, :]
```

```
            # chage error rate to attr + total
            Q_error_rate =
                np.append(Q_error_rate,
                np.append(np.apply_along_axis(
                np.sum, 0, q_diff != 0)/q_diff.shape[0],
                np.sum(q_diff != 0)/(
                    q_diff.shape[0]*q_diff.shape[1])
                ))
            #print('Q-matrix error rate: ', Q_error_rate[-1])
        #print(accuracy)
        #print(np.apply_along_axis(np.mean, 0, d))



print("Pattern Estimation accuracy:",
    np.mean(pattern_accuracy))
print("Attribute Estimation accuracy:",
    np.apply_along_axis(np.mean, 0,
        np.reshape(attribute_accuracy, [-1,n_attr])))
print("Q_matrix error rate:",
    np.mean(np.reshape(Q_error_rate, [-1,4])[:, -1]))

np.savetxt(
    file_path + 'pattern_accuracy.csv',
    pattern_accuracy,
    delimiter=',')
np.savetxt(
    file_path + 'attribute_accuracy.csv',
    np.reshape(attribute_accuracy, [-1,3]),
    delimiter= ',')
np.savetxt(
    file_path + 'Q_error_rate.csv',
        np.reshape(Q_error_rate, [-1,4]),
        delimiter=',')
```

Appendix Y

Python Code of DFN in Chapter 4

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 25 20:47:05 2019

@author: kangxue
"""

from keras.layers import Input, Dense,
    Dropout, Concatenate, Lambda, Add
from keras.models import Model
from keras import regularizers, optimizers,
    callbacks, initializers
from keras.constraints import min_max_norm,
    unitnorm, NonNeg
from keras.backend import slice
import tensorflow as tf


from sklearn.preprocessing import MinMaxScaler,
    LabelBinarizer
from sklearn.cluster import KMeans

import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
```

```
from sys import platform
import timeit

files = os.listdir(Data_path)
subfiles = '3_20_m_90'

true_label_df =
    pd.read_csv(os.path.join(Data_path,
    subfiles, "true_label.csv"))
DCM_attr_df =
    pd.read_csv(os.path.join(Data_path,
    subfiles, "error_label.csv"))
response_df =
    pd.read_csv((os.path.join(Data_path,
    subfiles, "response.csv")))
q_mat =
    np.array(pd.read_csv(os.path.join(Data_path,
    subfiles, 'q_mat.csv')))

n_attr = q_mat.shape[1]
n_item = q_mat.shape[0]

# convert to numpy array

true_attr = np.array(true_label_df)[:,0:n_attr]
DCM_attrs = (np.array(DCM_attr_df) >= 0.5)*1
response = np.array(response_df)*1


# functions for create labels
def get_labels(attr_est):
    var = np.array([], dtype='int')
    for i in range(attr_est.shape[1]):
        var = np.append(arr = var, values=2**i)
    return np.matmul(attr_est,
        var.reshape((-1, 1))).reshape((-1,1))+1

true_label = get_labels(true_attr).reshape((1,-1))[0]
np.mean(get_labels(DCM_attrs[:, 6:9]) == get_labels(true_attr))

# determine the all 1 group and all 0 group
def get_2_groups(groups, response):
```

```
    avg_score = np.array([])
    for group in np.unique(groups):
        score = np.mean(response[groups == group, :])
        avg_score = np.append(avg_score, score)

    return groups == np.argmax(avg_score)+1,
    groups == np.argmin(avg_score)+1

# calibrate kmeans to true labels
def cal_kmeans(km_label, true_label):
    km_label_2 = km_label.copy()
    for label in np.unique(km_label):
        t_label, t_count =
        np.unique(true_label[km_label == label],
            return_counts=True)
        km_label_2[km_label == label] =
            t_label[np.argmax(t_count)]

    return km_label_2

for i in range(5):
    DCM_attr = DCM_attrs[:, n_attr*i:n_attr*(i+1)]
    if i == 0:
        DCM_labels = get_labels(DCM_attr)
    else:
        DCM_labels =
            np.hstack((DCM_labels, get_labels(DCM_attr)))


LCDM_1, LCDM_0 = get_2_groups(DCM_labels[:, 2], response)
GDINA_1, GDINA_0 = get_2_groups(DCM_labels[:, 3], response)

# clustering KMeans
km = KMeans(n_clusters=2**n_attr)
km_label = (km.fit(response).labels_+1)

km_label_3 = cal_kmeans(km_label, true_label)

km_1, km_0 = get_2_groups(km_label, response)

labelbinarizer = LabelBinarizer()
km_label_1 =
```

```
    labelbinarizer.fit_transform(

np.random.seed(1234)
index = np.random.permutation(len(response))
x_train = response[index[:800], :]
y_train = km_label_1[index[:800], :]
x_test = response[index[800:],:]
y_test = km_label_1[index[800:], :]

regularizer_h1 = [1e-5, 1e-10, 1e-15]
Iter_max = 10
#iteration = 0


for iteration in range(0, Iter_max):
    inputs = Input(shape = (x_train.shape[1],))
    # first sub net
    hidden_1 =
        Dense(units= 200, activation='relu',  #120
              activity_regularizer =
              regularizers.l1(regularizer_h1[0]))(inputs)
    #drop1 = Dropout(.1)(hidden_1)
    hidden_2 =
        Dense(units= 100, activation='relu', # 60
              activity_regularizer =
              regularizers.l1(regularizer_h1[0]))(hidden_1)

    true_label = Dense(units = 2**n_attr,
        activation='softmax')(hidden_2)


    reconstruction =
        Dense(x_train.shape[1], activation = 'linear',
        kernel_constraint = min_max_norm(min_value = 0,
            max_value = 0.7),
        use_bias = False,
        kernel_initializer = initializers.Constant(value=0.4)
        )(true_label)

    # setup whole network
    deepFowardFeedNets =
        Model(inputs, [true_label,
```

```
    reconstruction]) # whole network

earlyStopping =
    callbacks.EarlyStopping(monitor='val_loss',
        patience=2, verbose=0, mode='auto')

deepFowardFeedNets.compile(optimizer=optimizers.Adam(),
                           loss=['categorical_crossentropy',
                                 'binary_crossentropy'],
                           loss_weights = [3, 5],
                           metrics=['accuracy'])

max_epochs = 3000
batch_size = 100

starting = timeit.default_timer()
history =
    deepFowardFeedNets.fit(x_train, [y_train, x_train],
                           epochs=max_epochs,
                           batch_size=batch_size,
                           shuffle=True,
                           verbose = 0,
                           validation_data=
                            (x_test,[y_test,x_test]),
                           callbacks=[earlyStopping])

print(timeit.default_timer() - starting)

dfn_label = n
    p.array(deepFowardFeedNets.predict_on_batch(response)[0])

dfn_label =
    np.apply_along_axis(func1d=np.argmax, axis=1,
        arr=dfn_label)+1

print(np.mean(dfn_label == true_label))

if iteration == 0:
    dfn_label_iteration_1, dfn_label_iteration_0 =
    get_2_groups(groups=dfn_label, response=response)
else:
    dfn_label_iteration_1 =
```

```
            np.vstack((dfn_label_iteration_1,
                get_2_groups(groups=dfn_label,
                    response=response)[0])
                    )
        dfn_label_iteration_0 =
            np.vstack((dfn_label_iteration_0,
                get_2_groups(groups=dfn_label,
                    response=response)[1])
                    )


dfn_1 =
    np.apply_along_axis(arr=dfn_label_iteration_1,
        axis=0, func1d=np.mean) > .5
dfn_0 =
    np.apply_along_axis(arr=dfn_label_iteration_0,
        axis=0, func1d=np.mean) > .5

print('Kmeans:',
    np.mean(km_1 == (true_label == 8)),
    np.mean(km_0 == (true_label == 1)))
print('DFN:',
    np.mean(dfn_1 == (true_label == 8)),
    np.mean(dfn_0 == (true_label == 1)))
print('LCDM:',
    np.mean(LCDM_1 == (true_label == 8)),
    np.mean(LCDM_0 == (true_label == 1)))
print('GDINA:',
    np.mean(GDINA_1 == (true_label == 8)),
    np.mean(GDINA_0 == (true_label == 1)))
```

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Et al. (2016). Tensorflow: A system for large-scale machine learning, In *12th {usenix} symposium on operating systems design and implementation ({osdi} 16)*.

Abedi, M., Sun, B., & Zheng, Z. (2019). A sinusoidal-hyperbolic family of transforms with potential applications in compressive sensing. *IEEE Transactions on Image Processing*, *28*(7), 3571–3583.

Aiserman, M., Braverman, È. M., & Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition. *Avtomat. i Telemeh*, *25*, 917–936.

Barron, A. R. (1994). Approximation and estimation bounds for artificial neural networks. *Machine learning*, *14*(1), 115–133.

Basilevsky, A. (1994). Statistical factor analysis and related methods. hoboken. *NJ, USA: John Wiley & Sons, Inc. DOI*, *10*, 9780470316894.

Beal, M. J., Ghahramani, Z., & Rasmussen, C. E. (2002). The infinite hidden markov model, In *Advances in neural information processing systems*.

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning, In *Proceedings of icml workshop on unsupervised and transfer learning*.

Bengio, Y., Laufer, E., Alain, G., & Yosinski, J. (2014). Deep generative stochastic networks trainable by backprop, In *International conference on machine learning*.

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

Bradshaw, L., Izsák, A., Templin, J., & Jacobson, E. (2014). Diagnosing teachers' understandings of rational numbers: Building a multidimensional test within the diagnostic classification framework. *Educational measurement: Issues and practice*, *33*(1), 2–14.

Briggs, D. C., & Circi, R. (2017). Challenges to the use of artificial neural networks for diagnostic classifications with student test data. *International Journal of Testing*, *17*(4), 302–321.

Britain, S., & Liber, O. (2004). A framework for pedagogical evaluation of virtual learning environments.

Brusco, M. J., Shireman, E., & Steinley, D. (2017). A comparison of latent class, k-means, and k-median methods for clustering dichotomous data. *Psychological methods*, *22*(3), 563.

Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, *20*(3), 542–542.

Chen, Y., & Li, X. (2019). Exploratory data analysis for cognitive diagnosis: Stochastic co-blockmodel and spectral co-clustering, In *Handbook of diagnostic classification models*. Springer.

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. A., & Li, X. (2009). Cluster analysis for cognitive diagnosis: Theory and applications. *Psychometrika*, *74*(4), 633.

Chiu, C.-Y., Köhn, H.-F., Zheng, Y., & Henson, R. (2016). Joint maximum likelihood estimation for diagnostic classification models. *psychometrika*, *81*(4), 1069–1092.

Chollet, F., & Allaire, J. J. (2018). *Deep learning mit r und keras: Das praxishandbuch von den entwicklern von keras und rstudio*. MITP-Verlags GmbH & Co. KG.

Crocker, L., & Algina, J. (1986). *Introduction to classical and modern test theory.* ERIC.

Csáji, B. C. Et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, *24*(48), 7.

Cui, Y., Guo, Q., & Cutumisu, M. (2017). A neural network approach to estimate student skill mastery in cognitive diagnostic assessments.

Cui, Y., Gierl, M. J., & Chang, H.-H. (2012). Estimating classification consistency and accuracy for cognitive diagnostic assessment. *Journal of Educational Measurement*, *49*(1), 19–38.

Cui, Y., Gierl, M., & Guo, Q. (2016). Statistical classification for cognitive diagnostic assessment: An artificial neural network approach. *Educational Psychology*, *36*(6), 1065–1082.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Dahl, G. E., Sainath, T. N., & Hinton, G. E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout, In *2013 ieee international conference on acoustics, speech and signal processing*. IEEE.

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. (2011). The generalized dina model framework. *Psychometrika*, *76*(2), 179–199.

DeCarlo, L. T. (2011). On the analysis of fraction subtraction data: The dina model, classification, latent class sizes, and the q-matrix. *Applied Psychological Measurement*, *35*(1), 8–26.

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

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

Evans, B., & Ossorio, P. (2018). The challenge of regulating clinical decision support software after 21st century cures. *American journal of law & medicine*, *44*(2-3), 237–251.

Fergus, R., Weiss, Y., & Torralba, A. (2009). Semi-supervised learning in gigantic image collections, In *Advances in neural information processing systems*.

George, A. C., Robitzsch, A., Kiefer, T., Groß, J., & Ünlü, A. (2016). The r package cdm for cognitive diagnosis models. *Journal of Statistical Software*, *74*(2), 1–24.

Gierl, M. J., Cui, Y., & Hunka, S. (2008). Using connectionist models to evaluate examinees' response patterns to achievement tests. *Journal of Modern Applied Statistical Methods*, *7*(1), 19.

Gierl, M. J., Wang, C., & Zhou, J. (2008). Using the attribute hierarchy method to make diagnostic inferences about examinees' cognitive skills in algebra on the sat. *Journal of Technology, Learning, and Assessment*, *6*(6), n6.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Grandvalet, Y., & Bengio, Y. (2005). Semi-supervised learning by entropy minimization, In *Advances in neural information processing systems*.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks, In *2013 ieee international conference on acoustics, speech and signal processing*. IEEE.

Grohs, P., Perekrestenko, D., Elbrächter, D., & Bölcskei, H. (2019). Deep neural network approximation theory. *arXiv preprint arXiv:1901.02220*.

Haffari, G. R., & Sarkar, A. (2012). Analysis of semi-supervised learning with the yarowsky algorithm. *arXiv preprint arXiv:1206.5240*.

Hanin, B. (2019). Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, *7*(10), 992.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100–108.

Hartz, S. M. (2002). *A bayesian framework for the unified model for assessing cognitive abilities: Blending theory with practicality*. (Doctoral dissertation). ProQuest Information & Learning.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition, In *Proceedings of the ieee conference on computer vision and pattern recognition*.

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.

Hinton, G., Osindero, S., Welling, M., & Teh, Y.-W. (2006). Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science*, *30*(4), 725–731.

Hornik, K., Stinchcombe, M., White, H., & Auer, P. (1994). Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. *Neural Computation*, *6*(6), 1262–1275.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, *24*(6), 417.

Ide, H., & Kurita, T. (2017). Improvement of learning for cnn with relu activation by sparse regularization, In *2017 international joint conference on neural networks (ijcnn)*. IEEE.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, *31*(8), 651–666.

Joachims, T. (1999). Chapter making large-scale svm learning practical. *Advances in Kernel Methods: Support Vector Learning*, *11*.

Johnson, M. (2009). A note on the estimable attribute sets in cognitive diagnostic models. *Unpublished Manuscript*.

Jolliffe, I. T. (2002). Choosing a subset of principal components or variables. *Principal component analysis*, 111–149.

Junker, B. W., & Sijtsma, K. (2001). Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, *25*(3), 258–272.

Karaca, Y., & Hayta, Ş. (2016). Application and comparison of ann and svm for diagnostic classification for cognitive functioning. *Appl. Math. Sci*, *10*(64), 3187–3199.

Kohonen, T. (2001). *Self-organizing*. Springer Berlin.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks, In *Advances in neural information processing systems*.

Kunina-Habenicht, O., Rupp, A. A., & Wilhelm, O. (2012). The impact of model misspecification on parameter estimation and item-fit assessment in log-linear diagnostic classification models. *Journal of Educational Measurement*, *49*(1), 59–81.

Larochelle, H., & Bengio, Y. (2008). Classification using discriminative restricted boltzmann machines, In *Proceedings of the 25th international conference on machine learning*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object recognition with gradient-based learning, In *Shape, contour and grouping in computer vision*. Springer.

Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks, In *Workshop on challenges in representation learning, icml*.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, In *Proceedings of the 26th annual international conference on machine learning*.

Leighton, J. P., Gierl, M. J., & Hunka, S. M. (2004). The attribute hierarchy method for cognitive assessment: A variation on tatsuoka's rule-space approach. *Journal of educational measurement*, *41*(3), 205–237.

Levine, M. V. (1982). The trait in latent trait theory.

Li, H., & Suen, H. K. (2013). Constructing and validating a q-matrix for cognitive diagnostic analyses of a reading test. *Educational Assessment*, *18*(1), 1–25.

Liang, P. (2005). *Semi-supervised learning for natural language* (Doctoral dissertation). Massachusetts Institute of Technology.

Liou, C.-Y., Huang, J.-C., & Yang, W.-C. (2008). Modeling word perception using the elman network. *Neurocomputing*, *71*(16-18), 3150–3157.

Liu, R., Huggins-Manley, A. C., & Bradshaw, L. (2017). The impact of q-matrix designs on diagnostic classification accuracy in the presence of attribute

hierarchies. *Educational and psychological measurement*, *77*(2), 220–240.

Liu, Y., & Kirchhoff, K. (2014). Graph-based semi-supervised acoustic modeling in dnn-based speech recognition, In *2014 ieee spoken language technology workshop (slt)*. IEEE.

Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017). The expressive power of neural networks: A view from the width, In *Advances in neural information processing systems*.

Madison, M. J., & Bradshaw, L. P. (2015). The effects of q-matrix design on classification accuracy in the log-linear cognitive diagnosis model. *Educational and Psychological Measurement*, *75*(3), 491–511.

Manning, C. D., Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Martınez-Plumed, F., Prudêncio, R. B., Martınez-Usó, A., & Hernández-Orallo, J. (2016). Making sense of item response theory in machine learning, In *Proceedings of the twenty-second european conference on artificial intelligence*. IOS Press.

Matsugu, M., Mori, K., Mitari, Y., & Kaneda, Y. (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, *16*(5-6), 555–559.

Miotto, R., Li, L., Kidd, B. A., & Dudley, J. T. (2016). Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, *6*(1), 1–10.

Mnih, V., & Hinton, G. E. (2012). Learning to label aerial images from noisy data, In *Proceedings of the 29th international conference on machine learning (icml-12)*.

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

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines, In *Proceedings of the 27th international conference on machine learning (icml-10)*.

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training, In *Proceedings of the ninth international conference on information and knowledge management*.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*, *45*(2), 227–248.

Paulsen, J. (2019). *Examining cognitive diagnostic modeling in small sample contexts* (Doctoral dissertation). Indiana University.

Rokach, L., & Maimon, O. (2005). Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *35*(4), 476–487.

Rosenberg, C., Hebert, M., & Schneiderman, H. (2005). Semi-supervised self-training of object detection models. *WACV/MOTION*, *2*.

Roussos, L., Henson, R., & Jang, E. (2005). Simulation study evaluation of the fusion model system stepwise algorithm. *ETS Project Report. Princeton, NJ: Educational Testing Service*.

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

Rupp, A. A., & Templin, J. L. (2007). Unique characteristics of cognitive diagnosis models, In *Annual meeting of the national council on measurement in education, chicago, il*.

Salakhutdinov, R., & Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, *50*(7), 969–978.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, *10*(5), 1299–1319.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions, In *Proceedings of the ieee conference on computer vision and pattern recognition*.

Taghipour, K., & Ng, H. T. (2016). A neural approach to automated essay scoring, In *Proceedings of the 2016 conference on empirical methods in natural language processing*.

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. (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.

Templin, J., & Bradshaw, L. (2013). Measuring the reliability of diagnostic classification model examinee estimates. *Journal of Classification*, *30*(2), 251–275.

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

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *61*(3), 611–622.

Tjoe, H., & de la Torre, J. (2014). The identification and validation process of proportional reasoning attributes: An application of a cognitive diagnosis modeling framework. *Mathematics Education Research Journal*, *26*(2), 237–255.

Touretzky, D., & Hinton, G. (1989). Connectionist models summer school, In *Proc. 1988 connectionist models summer school*. Morgan Kaufmann.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders, In *Proceedings of the 25th international conference on machine learning*.

von Davier, M. (2005). A general diagnostic model applied to language testing data. *ETS Research Report Series*, *2005*(2), i–35.

von Davier, M. (2018). Automated item generation with recurrent neural networks. *psychometrika*, *83*(4), 847–857.

Xu, G., & Zhang, S. (2016). Identifiability of diagnostic classification models. *Psychometrika*, *81*(3), 625–649.

Xue, K. (2018). Non-model based attribute profile estimation with partial q-matrix information for cognitive diagnosis using artificial neural network, In *Proceedings of the 11th international conference on educational data mining*.

Xue, K. (2019). Computational diagnostic classification model using deep feedforward network based semi-supervised learning, In *Proceedings of the 25th acm sigkdd conference on knowledge discovery and data mining (kdd) workshop on deep learning for education*.

Xue, K., Yaneva, V., Runyon, C., & Baldwin, P. (2020). Predicting the difficulty and response time of multiple choice questions using transfer learning, In *Proceedings of the 15th workshop on innovative use of nlp for building educational applications*.

Yamaguchi, K., & Okada, K. (2018). Comparison among cognitive diagnostic models for the timss 2007 fourth grade mathematics assessment. *PloS one*, *13*(2).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks, In *European conference on computer vision*. Springer.

Zell, A. (1994). *Simulation neuronaler netze* (Vol. 1). Addison-Wesley Bonn.

Zhu, X. J. (2005). *Semi-supervised learning literature survey* (tech. rep.). University of Wisconsin-Madison Department of Computer Sciences.

Zhu, X., & Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, *3*(1), 1–130.