

DEVELOPING MACHINE LEARNING ALGORITHMS FOR APPLICATION IN  
EDUCATIONAL MEASUREMENT

by

Geoffrey Converse

A thesis submitted in partial fulfillment of the  
requirements for the Doctor of Philosophy  
degree in Applied Mathematical and Computational Sciences  
in the Graduate College of  
The University of Iowa

Date?

Thesis Committee: Professor Suely Oliveira, Thesis Supervisor

Member Two

Member Three

Member Four

Member Five

Copyright by  
GEOFFREY CONVERSE  
2021  
All Rights Reserved

## ACKNOWLEDGEMENTS

## ABSTRACT

## PUBLIC ABSTRACT

# TABLE OF CONTENTS

LIST OF TABLES . . . . .	vii
LIST OF FIGURES . . . . .	viii
CHAPTER	
1 ITEM RESPONSE THEORY BACKGROUND . . . . .	2
1.1 Item Response Theory . . . . .	3
1.1.1 Rasch Model . . . . .	4
1.1.2 Normal Ogive Model . . . . .	6
1.1.3 2-Parameter Logistic Model . . . . .	6
1.1.4 Multidimensional Item Response Theory . . . . .	7
1.2 IRT Parameter Estimation . . . . .	8
1.2.1 Maximum Likelihood Estimation . . . . .	8
1.2.2 Joint Maximum Likelihood Estimation . . . . .	8
1.2.3 Marginal Maximum Likelihood Estimation . . . . .	8
1.3 Artificial Neural Networks . . . . .	8
1.3.1 Autoencoders . . . . .	10
1.3.2 Variational Autoencoders . . . . .	11
2 ESTIMATING IRT PARAMETERS VIA VARIATIONAL AUTOEN- CODERS . . . . .	12
2.1 ML2P-VAE Method Description . . . . .	12
2.1.1 Full Covariance Matrix Implementation . . . . .	14
2.1.2 Variants - 1PL and 3PL . . . . .	17
2.1.3 On Convergence . . . . .	17
2.1.3.1 Proving local minimum at true solution . . . . .	17
2.1.3.2 Requirements on sparsity of Q-matrix . . . . .	20
2.2 <b>ML2Pvae</b> Software Package for R . . . . .	21
2.2.1 Package Functionality . . . . .	21
3 ML2P-VAE RESULTS AND DISCUSSION . . . . .	23
3.1 Description of Data Sets . . . . .	23
3.2 Quantitative Results . . . . .	25
3.2.1 Preliminary Results . . . . .	25
3.2.2 Ablation Study . . . . .	25
3.2.3 Variational Autoencoder vs Autoencoder . . . . .	25

3.2.4	ML2P-VAE vs Traditional Methods . . . . .	29
4	KNOWLEDGE TRACING BACKGROUND . . . . .	31
4.1	Mathematical Setup . . . . .	32
4.2	Literature Review . . . . .	32
4.2.1	Deep Knowledge Tracing . . . . .	32
4.2.2	Dynamic Key-Value Memory Networks . . . . .	32
4.2.2.1	Deep-IRT . . . . .	32
4.2.3	Self-Attentive Knowledge Tracing . . . . .	32
4.2.4	Performance Factors Analysis . . . . .	33
5	DEEP, INTERPRETABLE METHODS FOR KNOWLEDGE TRACING . . . . .	34
5.1	Deep Performance Factors Analysis . . . . .	34
5.2	IRT Parameter Recovery + Knowledge Tracing . . . . .	34
6	KNOWLEDGE TRACING RESULTS . . . . .	35
6.1	Data Description . . . . .	35
6.2	Experiment Details . . . . .	35
6.3	Results . . . . .	35
6.3.1	With Transformers . . . . .	35
6.3.2	Deep PFA . . . . .	35
6.3.3	IRT parameter recovery + KT . . . . .	35
7	RELATED WORK . . . . .	36
7.1	Health Sciences . . . . .	36
7.2	Sports Analytics . . . . .	36
7.2.1	VAE for Player Evaluation . . . . .	37
7.2.2	Experiments . . . . .	38
	REFERENCES . . . . .	41

# LIST OF TABLES

Table

3.1	Statistics for item parameter recovery. . . . .	26
3.2	Statistics for latent trait prediction. . . . .	28
3.3	Error measures for discrimination ( $a$ ), difficulty ( $b$ ), and ability ( $\theta$ ) parameters from various parameter estimation methods on three different data sets. Note that in the ECPE data set, there are no true values, so MHRM estimates are accepted as true. In Sim-20, only ML2P-VAE methods are capable of estimating such high-dimensional latent traits . . . . .	30



# LIST OF FIGURES

Figure

1.1	An item characteristic curve visualizes the relation between a student's ability and the probability of answering an item correctly. . . . .	5
3.1	Autoencoder and VAE discrimination parameter ( $a_{ji}$ ) recovery. . . . .	27
3.2	Autoencoder and VAE difficulty parameter ( $b_i$ ) recovery. . . . .	27
3.3	Autoencoder and VAE predictions for $\theta_1$ . . . . .	29
7.1	Each latent skill's estimates plotted against its evaluation statistic. . . .	40

## INTRODUCTION

This thesis is organized in three parts. In Part I, the first three chapters introduce Item Response Theory (IRT) and analyzes the novel parameter estimation method, ML2P-VAE. This method uses a modified variational autoencoder to estimate parameters in IRT models. Chapters 4-6 of Part II explore a task common in electronic learning environments: knowledge tracing. While other deep learning methods for knowledge tracing lack interpretability, new methods presented here present a trade-off between prediction power and explainability. Part III explores two applications of the ML2P-VAE method in areas outside of education: sports analytics and health sciences.

## CHAPTER 1

### ITEM RESPONSE THEORY BACKGROUND

In educational measurement, a common goal is to quantify the knowledge of students from the results of some assessment. In a classroom setting, grades are typically assigned based on the percentage of questions answered correctly by a student assignments. The letter grades assigned from these percentages can serve as a naive measure of student knowledge; “A” students have completely mastered the material, “B” students have a good grasp of material, “C” students are fairly average, and “D” and “F” students have significant gaps in their knowledge.

The practice of evaluating student ability purely from a raw percentage score is known as true score theory [19]. But there are clear issues with this approach. Not all questions on an exam or homework assignment is created equally: some questions are easier, and some more difficult. Consider a scenario where two students both answer 17 out of 20 questions correctly on a test for a raw score of 85%. But if Student A answered questions 1, 8, and 9 wrong while Student B answered 4, 17, and 20 incorrectly, it is not likely that that Student A and Student B possess the same level of knowledge. For example, questions 1, 8, and 9 could be much more difficult than questions 4, 17, and 20. Additionally, the two sets of problems could cover different types of material. True score theory does not account for either of these situations, and naively quantifies the knowledge of Student A and Student B as equal.

More sophisticated methods have been studied which attempt to more ac-

curately quantify student learning. Cognitive Diagnostic Models (CDM) (TODO: citation) aim to classify whether students possess mastery of a given skill or not. This discrete classification can be useful in determining whether or not a student meets a prerequisite, or deciding whether or not they are ready to move on to the next level of coursework. We focus instead on Item Response Theory, where student knowledge is assumed to be continuous.

### 1.1 Item Response Theory

Item Response Theory (IRT) is a field of quantitative psychology which uses statistical models to model student ability [14]. These models often give the probability of a question being answered correctly as a function of the student’s ability. In IRT, it is assumed that each student, indexed by  $j$ , possesses some continuous latent ability  $\theta_j$ . The term “latent ability” is synonymous with “knowledge” or “skill.” Often, it is assumed that amongst the population of students,  $\theta_j \sim \mathcal{N}(0, 1)$  [19].

In this work, we often consider the case where each student has multiple latent abilities. For example, in the context of an elementary math exam, we may wish to measure the four distinct skills “add”, “subtract”, “multiply”, and “divide.” This scenario is referred to as multidimensional item response theory, and we write the set of student  $j$ ’s  $K$  latent abilities as a vector  $\Theta_j = (\theta_{1j}, \theta_{2j}, \dots, \theta_{Kj})^\top$ . It is then assumed that the latent abilities of students follow some multivariate Gaussian distribution,  $\mathcal{N}(0, \Sigma)$ . For simplicity, the covariance matrix  $\Sigma$  is often taken to be the identity matrix, making each latent skill independent of one another.

Note that  $\Theta_j$  is not directly observable in any way. Instead, a common goal is to infer student's knowledge  $\Theta_j$  from on their responses on some assessment containing  $n$  questions, referred to as items. A student's set of responses can be written as a binary  $n$ -dimensional vector  $\vec{u}_j = (u_{1j}, u_{2j}, \dots, u_{nj})^\top$ , where

$$u_{ij} = \begin{cases} 1 & \text{if student } j \text{ answers item } i \text{ correctly} \\ 0 & \text{otherwise} \end{cases} \quad (1.1)$$

IRT models aim to model the probability of a student answering a particular question correctly, so that the probability of student  $j$  answering item  $i$  correctly is given by some function of  $\Theta_j$ :

$$P(u_{ij} = 1 | \Theta_j) = f(\Theta_j; V_i) \quad (1.2)$$

where  $V_i$  is a set of parameters associated with item  $i$ . In general,  $f : \mathbb{R}^K \rightarrow [0, 1]$  is some continuous function which is strictly increasing with respect to  $\Theta_j$ .

In the following sections, we describe various candidates for the function  $f$ . Though each is presented in the context of single-dimensional IRT ( $K = 1$ ), they can all be easily adapted to higher dimensions.

### 1.1.1 Rasch Model

One of the first models was proposed by Georg Rasch in 1960. Rasch asserted that the probability of a student answering an item correctly is a function of the ratio  $\xi/\delta$ , where  $\xi > 0$  represents the student's knowledge, and  $\delta > 0$  quantifies the difficulty of an item. Consider the  $\frac{\xi}{\xi+\delta} = \frac{1}{1+\delta/\xi}$  and note that  $\frac{\xi}{\xi+\delta} \rightarrow 1$  as  $\xi \rightarrow \infty$ . After the reparametarization  $\xi = e^\theta$  and  $\delta = e^b$ , we arrive at the 1-Parameter Logistic



Figure 1.1: An item characteristic curve visualizes the relation between a student’s ability and the probability of answering an item correctly.

Model, often referred to as the Rasch Model.

$$P(u_{ij} = 1|\theta_j; b_i) = \frac{1}{1 + e^{b_i - \theta_j}} \quad (1.3)$$

Note that  $\theta \in \mathbb{R}$  and  $b \in \mathbb{R}$  still represent student ability and item difficulty, respectively. We can interpret the difficulty parameter  $b$  as a threshold: when  $\theta = b$ , then the student has a 50% chance of answering the question correctly. A plot of Equation 1.3 for a fixed item (fixed  $b_i$ ) is shown in Figure 1.1. The horizontal axis represents  $\log \theta$ , and the vertical axis represents  $P(u_{ij} = 1|\theta, b_i)$ . This type of graph is often referred to as an item characteristic curve (ICC).

### 1.1.2 Normal Ogive Model

A slightly more sophisticated method for measuring student performance is the normal ogive model. We introduce a discrimination parameter,  $a_i$ , which quantifies the capability of item  $i$  in distinguishing between students who have / have not mastered the knowledge concept  $\theta$  [19]. In other words,  $a_i$  tells *how much* of skill  $\theta$  is required to answer item  $i$  correctly.

The normal ogive model give the probability of student  $j$  answering item  $i$  correctly as

$$P(u_{ij} = 1|\theta_j; a_i, b_i) = \frac{1}{\sqrt{2\pi}} \int_{-a_i\theta_j+b_i}^{\infty} e^{-\frac{z^2}{2}} dz \quad (1.4)$$

Note the similarity between Equation 1.4 and the cumulative distribution function for a Gaussian distribution. The normal ogive model is popular among statisticians for this reason, but can be difficult to use for parameter estimation.

### 1.1.3 2-Parameter Logistic Model

The model which this work focuses on most is the 2-parameter logistic (2PL) model. Like the normal ogive model, the 2PL model uses both the discrimination and difficulty item parameters. The probability of student  $j$  answering item  $i$  correctly is given by

$$P(u_{ij} = 1|\theta_j; a_i, b_i) = \frac{1}{1 + e^{-a_i\theta_j+b_i}} \quad (1.5)$$

Equation 1.5 has the same form as that of the Rasch model in Equation 1.3, but adds in the discrimination parameter  $a_i$ . If this parameter is scaled by 1.7, then the ICC from the normal ogive model differs from that of the 2PL model by 0.01 [1].

\*This section  
isn't very  
important to  
rest of work

In a sense, we can consider the 2PL model to be a very good approximation of the normal ogive model. Due to the simple form of Equation 1.5, using this model makes parameter estimation much easier.

#### 1.1.4 Multidimensional Item Response Theory

The previously described statistical models can all be extended so that each student possesses  $K$  latent traits. In multidimensional item response theory (MIRT), models give the probability of a correct answer as a function of the student ability vector  $\Theta = (\theta_1, \dots, \theta_K)^\top$ . The generalization of 1.5 is given by the multidimensional logistic 2-parameter (ML2P) model:

$$P(u_{ij} = 1 | \Theta_j; \vec{a}_i, b_i) = \frac{1}{1 + \exp(-\vec{a}_i^\top \Theta_j + b_i)} = \frac{1}{\exp\left(-\sum_{k=1}^K a_{ik}\theta_{kj} + b_i\right)} \quad (1.6)$$

Here, the discrimination parameters  $\vec{a}_i \in \mathbb{R}^K$  are given as vector, where each entry  $a_{ik} \in \vec{a}_i$  quantifies *how much* of skill  $k$  is required to answer item  $i$  correctly. The ML2P model is the main focus of this thesis.

TODO: mention MDISC and how this scales

In MIRT, it is convenient to notate the relationship between skills and items with binary matrix. Define the  $Q$ -matrix [8]  $Q \in \{0, 1\}^{n \times K}$  so that

$$q_{ik} = \begin{cases} 1 & \text{if item } i \text{ requires skill } k \\ 0 & \text{otherwise} \end{cases}. \quad (1.7)$$

In real applications, the  $Q$ -matrix is annotated by an expert in the field, as it is usually possible to discern the concepts need to answer an item correctly. In relation to the ML2P model (Equation 1.6), notice that if  $q_{ik} = 0$ , then  $a_{ik} = 0$  as well. Though



experts can produce a  $Q$ -matrix for a given assessment, the matrix of discrimination parameters  $(a_{ik})_{i,k}$  can not be discovered so easily.

\*How in-depth  
do these  
descriptions  
need to be?  
Could just give  
overall idea  
and describe  
weaknesses

## 1.2 IRT Parameter Estimation

### 1.2.1 Maximum Likelihood Estimation

TODO: item parameter estimation

TODO: ability parameter estimation

### 1.2.2 Joint Maximum Likelihood Estimation

### 1.2.3 Marginal Maximum Likelihood Estimation

TODO: MMLE

TODO: EM

## 1.3 Artificial Neural Networks

In recent years, artificial neural networks (ANN) have become an increasingly popular tool for machine learning problems. Though they have been around since the 1960's (TODO: citation), GPU technology has become more accessible and modern computers are more powerful, allowing anyone interested to train a basic neural network on their machine. ANN can be applied to a diverse set of problems, including regression, classification, computer vision, natural language processing, function approximation, data generation, and more (TODO: citations).

One of the biggest critiques of ANN is their black-box nature, meaning that the

decision process that a trained model uses is typically not explainable by humans. As opposed to simpler methods such as decision trees or linear regression, neural networks are not interpretable. This makes them less desirable in certain applications where researchers wish to know *why* a model predicts a particular data sample the way that it does. For example, if a financial institution is using data science methods to determine whether or not to approve someone’s loan, the institution should be able to explain to the customer why they were denied. Most customers will not be satisfied with “the computer told us so,” and there is a possibility that a black-box neural network could learn and use features such as race or gender in its prediction, which is illegal in the United States (TODO: definitely need citation or delete).

The push for explainable AI have led researchers down two paths. One group has tried to incorporate deep learning methods with existing interpretable methods, in hopes of increasing the performance of explainable methods without sacrificing its interpretability (TODO: citation). Another option is to use a sort of hybrid learning, where interpretable models defer to a black-box model if they are not confident in their prediction [16]. Others have started with deep models and cut back on complexity, making specific modifications which increase interpretability. For example, the loss function of a convolutional neural network can be adapted so that humans can understand the features extracted in the hidden layers [22].

The field of education is an application which often desires interpretable models. Researchers often need to be able to point out specific details of decisions made by AI. A student deserves an answer to *why* they failed a test, and a teacher should

be given instructions on *how* to fix the student's misconceptions.

### 1.3.1 Autoencoders

An autoencoder (AE) is a neural network where the input and output layers are the same shape. The objective for a given data point is to minimize the difference between the output, called the reconstruction, and the input. Typically, the middle hidden layers of an AE are of smaller dimension than the input space. In this way, autoencoders are an unsupervised learning technique for (nonlinear) dimension reduction. Mathematically, we can define an autoencoder in two parts as follows.

For an input  $x \in \mathbb{R}^n$ , define the *encoder* as a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  mapping  $x \mapsto z := f(x)$ . Usually,  $m < n$ , and  $z$  lies in a hidden feature space. The encoder sends an observed data point to its representation in a learned feature space. Define the *decoder* as a function  $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$  mapping  $z \mapsto \hat{x} := g(z)$ . The decoder maps a hidden representation  $z$  to a reconstruction of the encoder input. Note that in our case, the functions  $f$  and  $g$  are both parameterized by neural networks, each of which can have any number of hidden layers. The end-to-end autoencoder is then the function composition  $\mathcal{A}(x) := g(f(x)) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ . To train an AE, the loss function minimizes the difference between the input and output. This can be done in a number of ways, including the simple mean squared error loss

$$\mathcal{L}(x) = \|x - g(f(x))\|_2^2 \quad (1.8)$$

or cross-entropy loss for binary data

$$\mathcal{L}(x) = \sum_{i=1}^n -x_i \log(g(f(x_i))) - (1 - x_i) \log(1 - g(f(x_i))). \quad (1.9)$$

Autoencoders with only a single hidden layer can be compared with nonlinear principal components analysis (PCA), and using linear activation functions allows for recovery of PCA loading vectors [15]. AEs have clear applications in image compression straight out-of-the-box, and can be modified for more complicated problems. Denoising autoencoders [20] are capable of processing noisy images and cleaning them up. To do this, they corrupt input data by deleting pixels at random and reconstructing the original image. Autoencoders can also be modified for data generation applications using a variational autoencoder.

### 1.3.2 Variational Autoencoders

\*relevant sources: [9] [13] [2], infoVAE, ELBO, “towards deeper understanding of VAE”

TODO: describe probabilistic derivation of VAE (ie Kingma and Welling). Also talk about how Zhao et al (InfoVAE) show that if decoder is Gaussian, then maximizing ELBO makes the latent distribution bad - but I’ve shown this isn’t the case in our model, where the decoder is Bernoulli.

## CHAPTER 2

### ESTIMATING IRT PARAMETERS VIA VARIATIONAL AUTOENCODERS

The primary contribution of this thesis is the development of a method for IRT parameter estimation which uses a modified VAE. The method, titled “ML2P-VAE”, is interesting from multiple perspectives. In the application area, it is an unconventional approach can produce estimates as accurate as those of traditional parameter estimation techniques. Further, ML2P-VAE scales much better than traditional methods as the number of latent abilities becomes large. In the field of machine learning, ML2P-VAE is an unsupervised learning technique which yields explainable results. A variational autoencoder is used in an unorthodox way, and the trainable parameters and a hidden neural layer are able to be understood in a real-world context.

#### 2.1 ML2P-VAE Method Description

Assume we are given an assessment with  $n$  items which tests  $K$  latent skills, and that  $N$  students take this exam. Data is given as an  $N \times n$  binary matrix, as in Equation 1.1. No information about student latent ability parameters  $\Theta \in \mathbb{R}^K$  or item parameters  $a_{ik}$  and  $b_i$  is provided. However, there is access to an expert-annotated binary  $Q$ -matrix detailing the item-skill associations, as in Equation 1.7.

A number of specifications are required in order to use a VAE as an IRT parameter estimation method. First, set up the neural network so that the input and output layers each have  $n$  nodes, each representing one item. The inputs are be

the binary response vectors  $\vec{u}_j$  and the outputs are approximations of the probability of student  $j$  answering each item correctly. Next, the dimension of the hidden distribution (the output of the encoder) must be equal to  $K$ , the number of latent skills.

The modifications to a typical VAE architecture are focused on the decoder. No hidden layers are used in the decoder. Instead, a non-dense set of weights connects the decoder input to the decoder output. The non-zero weights here are determined by the  $Q$ -matrix; recall that the input to the decoder has  $K$  nodes, the decoder output has  $n$  nodes, and  $Q \in \{0, 1\}^{n \times K}$ . Further, require the use of the sigmoidal activation function

$$\sigma(z_i) = \frac{1}{1 + e^{z_i}} \quad (2.1)$$

in the output layer. Here,  $z_i = \sum_{k=1}^K w_{ik} \alpha_k + \beta_i$ , where  $w_{ik}$  is the weight between the  $k$ -th and  $i$ -th nodes in the decoder input and decoder output layer,  $\alpha_k$  is the activation of the  $k$ -th node in the decoder input layer, and  $\beta_i$  is the additive bias in the output layer. Note the similarity between Equation 2.1 and Equation 1.6. The constraint on the weights along with the sigmoidal activation function allows for interpretation of the decoder as an ML2P model.

Specifically, the decoder weights  $w_{ik}$  can be interpreted as estimates to the discrimination parameters  $a_{ik}$ , the output bias  $\beta_i$  can be interpreted as estimates to the difficulty parameters  $b_i$ , and the activations  $\alpha_k$  produced by the encoder (given response input  $\vec{u}_j$ ) can be interpreted as estimates to the student ability parameter  $\theta_{kj}$ .

may want to  
mention that  
the value  
outputted by  
encoder is then  
sampled from  
for decoder  
input

Further modifications can improve the performance of ML2P-VAE. In IRT, discrimination parameters are assumed to be non-negative, because an increase in a skill should never decrease the probability of answering an item correctly. With this assumption in mind, requiring all decoder weights  $w_{ik} \geq 0$  avoids a potential identification problem.

explain what  
this means  
 $-\theta a_{ik} =$   
 $\theta(-a_{ik})$

### 2.1.1 Full Covariance Matrix Implementation

There are many publicly available code exmples of VAE implementations which assume the latent space follows a standard normal distribution  $\mathcal{N}(0, I)$ . But it is not so common to train a VAE which assumes the latent prior  $p(\Theta)$  has correlated dimensions. Since most applications do not attempt to interpret hidden layers of a VAE, there is no available information on the correlations of abstract, unobservable features. Additionally, it is often beneficial to force the latent dimensions to be independent of one another.

In IRT, we may be able to quantify the correlation between latent abilities, presenting need for the ML2P-VAE model to take advantage of this information. This task is nontrivial due to two mechanisms of the VAE:

- (1) sampling from the learned distribution, and
- (2) calculating Kullback-Liebler Divergence

These two characteristics must be addressed when constructing the neural architecture.

After training a VAE, sending a data point  $u_0$  through the encoder needs to

give a set of values that correpond to a probability distribution. For a  $K$ -dimensional multivariate Gaussian distribution, these values are a vectore  $\mu_0 \in \mathbb{R}^K$  and a symmetric, positive-definite matrix  $\Sigma_0 \in \mathbb{R}^{K \times K}$ . Sampling from  $\mathcal{N}(\mu_0, \Sigma_0)$  requires a matrix  $G_0$  such that  $G_0 G_0^\top = \Sigma_0$ . This matrix factorization  $G_0$  is not unique, but it can be convenient to use the Cholesky decomposition of  $\Sigma_0$  [?]. The sample of the multivariate Gaussian is calculated as  $z_0 = \mu_0 + G_0 \vec{\varepsilon}_0$ , where  $\vec{\varepsilon}_0 = (\varepsilon_1, \dots, \varepsilon_K)^\top$  and  $\varepsilon_i \sim \mathcal{N}(0, 1)$ .

The KL-Divergence between two  $K$ -variate Gaussian distributions is given as

$$\begin{aligned} \mathcal{D}_{KL} [\mathcal{N}(\mu_0, \Sigma_0) || \mathcal{N}(\mu_1, \Sigma_1)] = \\ \frac{1}{2} \left( \text{tr}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1} (\mu_1 - \mu_0) - K + \ln \left( \frac{\det \Sigma_1}{\det \Sigma_0} \right) \right) \end{aligned} \quad (2.2)$$

When using this in a VAE,  $\mathcal{N}(\mu_1, \Sigma_1)$  corresponds to the prior  $p(\Theta)$ , and so  $\mu_1$  and  $\Sigma_1$  are constant. Then  $\Sigma_1^{-1}$  only needs to be computed once, and this matrix inversion won't cause computation time problems at any point. Note that Equation 2.2 computes  $\ln \det \Sigma_0$ , so we must ahve  $\det \Sigma_0 > 0$  at any point during training. Recall that  $\mu_0$  and  $\Sigma_0$  correspond to the input  $u_0$ , and also depend on all the trainable weights and biases in the VAE encoder. These parameters are usually initialized randomly, and the user has little control over their values during training. If  $\det \Sigma_0 \leq 0$  for any input  $u_0$  at any point during training, then it is not possible to compute the loss and gradient. Thus, a specific architecture which guarantees that  $\det \Sigma_0 > 0$ , regardless of the input  $u_0$  or encoder parameters, is required.

This architecture is described as follows. The input and output to the neural network consists of  $n$  nodes, each representing an item on an assessment. After a



sufficient number of hidden layers of sufficient size, the encoder outputs  $K + \frac{K(K+1)}{2}$  nodes. The first  $K$  nodes represent the mean vector  $\mu_0$ , and the remaining  $\frac{K(K+1)}{2}$  nodes are arranged into a lower triangular matrix  $L_0 \in \mathbb{R}^{K \times K}$ . The covariance matrix is obtained by using the matrix exponential  $\Sigma_0 = e^{L_0} \cdot (e^{L_0})^\top$ .

**Theorem 2.1.**  *$\Sigma_0$  constructed as described previously is symmetric, positive-definite, and has positive determinant.*

*Proof.* Consider any lower triangular  $L_0 \in \mathbb{R}^{K \times K}$ . Define

$$G_0 = e^{L_0} = \sum_{n=1}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0 \cdot L_0 + \dots$$

$G_0$  is lower triangular, since addition and multiplication of matrices preserve this property. Further,  $G_0$  is nonsingular, because  $\det G_0 = \det(e^{L_0}) = e^{\text{tr} L_0} > 0$ .

Set  $\Sigma_0 = G_0 G_0^\top$ . Clearly,  $\Sigma_0$  is symmetric as  $\Sigma_0^\top = (G_0 G_0^\top)^\top = G_0 G_0^\top = \Sigma_0$ .

Further,  $\det \Sigma_0 = \det G_0 \cdot \det G_0^\top > 0$ . So now for any nonzero  $x \in \mathbb{R}^K$ ,

$$\langle \Sigma_0 x, x \rangle = x^\top \Sigma_0 x = x^\top G_0 G_0^\top x = \langle G_0^\top x, G_0^\top x \rangle = \|G_0 x\|_2^2 > 0$$

Therefore,  $\Sigma_0$  is positive-definite. □

Theorem 2.1 shows that in this specific neural network architecture,  $\Sigma_0$  can be interpreted as a covariance matrix. Thus, the VAE encoder maps a data point  $u_0$  to a multivariate Gaussian distribution  $\mathcal{N}(\mu_0, \Sigma_0)$ . Additionally, the sampling operation and KL-Divergence calculation can always be carried out without issue.

### 2.1.2 Variants - 1PL and 3PL

### 2.1.3 On Convergence

#### 2.1.3.1 Proving local minimum at true solution

We define the true and predicted probability of student  $j$  answering item  $i$  correctly with  $P_{ij}$  and  $\hat{P}_{ij}$ , respectively. The former comes from the ML2P model, and the latter is the output of a neural network. We assume the more simple case, where  $\Theta \sim \mathcal{N}(0, I)$ , rather than  $\mathcal{N}(\mu, \Sigma)$ .

$$\begin{aligned} P_{ij} &= \frac{1}{1 + \exp\left(-\sum_{k=1}^K a_{ik}\theta_{jk} + b_i\right)} \\ \hat{P}_{ij} &= \frac{1}{1 + \exp\left(-\sum_{k=1}^K \hat{a}_{ik}(\hat{\theta}_{jk} + \varepsilon_k \hat{\sigma}_k) + \hat{b}_i\right)} \end{aligned} \quad (2.3)$$

Note that  $P_{ij}$  is unknown, and we instead have a response sequence  $\vec{u}_j = (u_{1j}, \dots, u_{nj})^\top$  with  $u_{ij} = \text{Bern}(P_{ij})$ . This also means that  $\mathbb{E}[u_{ij}] = P_{ij}$ . The variables  $\hat{a}_{ik}$ ,  $\hat{b}_i$ ,  $\hat{\theta}_{ik}$ , and are parameter estimates from the VAE. The first two are parameters in the neural network, and the ability estimates are taken from feeding responses to the encoder, i.e.,  $\hat{\Theta}_j = \text{Encoder}(\vec{u}_j)$ . The noise  $\varepsilon = (\varepsilon_k)_{1 \leq k \leq K} \sim \mathcal{N}(0, I)$  is introduced by the sampling operation in the VAE.

The loss function for a VAE is given by

$$\begin{aligned} \mathcal{L}(\vec{u}_j) &= -\sum_{i=1}^n \left[ u_{ij} \log(\hat{P}_{ij}) + (1 - u_{ij}) \log(1 - \hat{P}_{ij}) \right] + \mathbb{E}_{q_\alpha(\hat{\theta}|\vec{u}_j)} \log \left( \frac{q_\alpha(\hat{\theta}|\vec{u}_j)}{p(\theta)} \right) \\ &= \mathcal{L}_{\text{REC}} + \mathcal{L}_{\text{KL}} \end{aligned} \quad (2.4)$$

Can't really  
guarantee  
convergence  
globally with  
SGD because  
neural nets are  
super  
nonconvex

Instead of VAE  
loss (which is  
really ELBO),  
this should be  
the marginal  
prob. of data  
(see King-  
ma/Welling)

We break up the VAE loss into two terms, the reconstruction loss  $\mathcal{L}_{\text{REC}}$  and the KL-divergence loss  $\mathcal{L}_{\text{KL}}$ . In the latter, the distribution  $q_\alpha(\hat{\theta}|\vec{u}_j)$  is the output of the encoder, and  $p(\theta)$  is the assumed prior distribution of  $\Theta$ , which we set to be  $\mathcal{N}(0, I)$ .

We write  $P_{ij} = \mathbb{E}(u_{ij})$ , and similarly define a new “expected” loss function:

$$\begin{aligned}\mathcal{L}_{\mathbb{E}}(P_j) &= \mathbb{E}_{u_j}[\mathcal{L}(u_{:j})] \\ &= -\sum_{i=1}^n [P_{ij} \log(\hat{P}_{ij}) + (1 - P_{ij}) \log(1 - \hat{P}_{ij})] + \mathbb{E}_{q_\alpha(\hat{\theta}|u_j)} \log \left( \frac{q_\alpha(\hat{\theta}|u_j)}{p(\theta)} \right) \\ &= \mathcal{L}_{\mathbb{E}[\text{REC}]} + \mathcal{L}_{\mathbb{E}[\text{KL}]}\end{aligned}\tag{2.5}$$

Notice that calculation of this “expected loss” requires the unknown  $P_{ij}$ . But when we have large amounts of data, we can think of  $P_{ij}$  as the average value of the response  $u_{ij}$ , so using this unknown value here is justified.

Define  $z_i = a_{i:} \cdot \theta_{j:} - b_i$  and  $\hat{z}_i = \hat{a}_{i:} \cdot (\hat{\theta}_{j:} + \varepsilon \cdot \hat{\sigma}) - \hat{b}_i$ . Note that  $z_i$  is fixed, dependent on the data, and does not depend on any parameters of the neural network.  $\hat{z}_i$  is the input to the final layer of the decoder, and the VAE output is  $\hat{P}_{ij} = \sigma(\hat{z}_i)$ , where  $\sigma(\cdot)$  is the sigmoidal activation function. We compute derivatives of the expected loss function, looking individually at the reconstruction and KL terms.

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]} }{\partial \hat{z}_i} &= \frac{-1}{1 + e^{-z_i}} \cdot \frac{1}{1 + e^{\hat{z}_i}} - \frac{1}{1 + e^{z_i}} \cdot \frac{-1}{1 + e^{-\hat{z}_i}} \\ &= \frac{-1}{(1 + e^{-z_i})(1 + e^{\hat{z}_i})} + \frac{1}{(1 + e^{z_i})(1 + e^{-\hat{z}_i})} \\ &= \frac{-1}{(1 + e^{-a_{i:} \cdot \theta_{j:} + b_i})(1 + e^{\hat{a}_{i:} \cdot (\hat{\theta}_{j:} + \varepsilon \cdot \hat{\sigma}) - \hat{b}_i})} \\ &\quad + \frac{1}{(1 + e^{a_{i:} \cdot \theta_{j:} - b_i})(1 + e^{-\hat{a}_{i:} \cdot (\hat{\theta}_{j:} + \varepsilon \cdot \hat{\sigma}) + \hat{b}_i})}\end{aligned}\tag{2.6}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{a}_{ik}} &= \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{z}_i} \frac{\partial \hat{z}_i}{\partial \hat{a}_{ik}} = \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{z}_i} (\hat{\theta}_{jk} + \varepsilon_k \hat{\sigma}_k) \\
\frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{b}_i} &= \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{z}_i} \frac{\partial \hat{z}_i}{\partial \hat{b}_i} = \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{z}_i} (-1) \\
\frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{\theta}_{ik}} &= \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{z}_i} \frac{\partial \hat{z}_i}{\partial \hat{\theta}_{ik}} = \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{z}_i} (\hat{a}_{ik})
\end{aligned} \tag{2.7}$$

Rather than setting these to zero and solving, we show that the most intuitive solution,  $\hat{a}_{ik} = a_{ik}$ ,  $\hat{b}_i = b_i$ , and  $\hat{\theta}_{jk} = \theta_{jk}$ , is in fact a minimum of the expected loss function. But first, we must take another expectation over the random variable  $\varepsilon \sim \mathcal{N}(0, I)$ . Obviously, we have that  $\mathbb{E}[\varepsilon_k] = 0$ ; this makes our calculations very simple. Notice that we have

$$\begin{aligned}
&\mathbb{E}_{\varepsilon} \left[ \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{z}_i} \right] \Big|_{\hat{a}_{ik}=a_{ik}, \hat{b}_i=b_i, \hat{\theta}_{jk}=\theta_{jk}} \\
&= \frac{-1}{(1 + e^{-a_{ik}\theta_{jk}+b_i})(1 + e^{a_{ik}(\theta_{jk}+0\cdot\hat{\sigma}_k)-b_i})} + \frac{1}{(1 + e^{a_{ik}\theta_{jk}-b_i})(1 + e^{-a_{ik}(\theta_{jk}+0\cdot\hat{\sigma}_k)+b_i})} \\
&= 0
\end{aligned} \tag{2.8}$$

Therefore we clearly have

$$\begin{aligned}
&\mathbb{E}_{\varepsilon} \left[ \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{a}_{ik}} \right] \Big|_{\hat{a}_{ik}=a_{ik}, \hat{b}_i=b_i, \hat{\theta}_{jk}=\theta_{jk}} \\
&= \mathbb{E}_{\varepsilon} \left[ \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{b}_i} \right] \Big|_{\hat{a}_{ik}=a_{ik}, \hat{b}_i=b_i, \hat{\theta}_{jk}=\theta_{jk}} \\
&= \mathbb{E}_{\varepsilon} \left[ \frac{\partial \mathcal{L}_{\mathbb{E}[\text{REC}]}}{\partial \hat{\theta}_{jk}} \right] \Big|_{\hat{a}_{ik}=a_{ik}, \hat{b}_i=b_i, \hat{\theta}_{jk}=\theta_{jk}} \\
&= 0 \quad \forall i, j, k
\end{aligned} \tag{2.9}$$

This proves that the true parameters give a local minimum for the expected reconstruction error in the VAE loss.

We now consider the Kullback-Leibler divergence term in the expected loss function. Again assuming independent latent traits, we have

$$\mathcal{L}_{KL} = \mathbb{E}_{q(\theta|u)} \log \left( \frac{q(\hat{\theta}|u)}{p(\theta)} \right) = KL(q(\hat{\theta}|u)||p(\theta)) = -\frac{1}{2} \sum_{k=1}^K (1 + \log(\hat{\sigma}_k^2) - \hat{\theta}_k^2 - \hat{\sigma}_k^2) \quad (2.10)$$

It is clear that this regularization term is minimized (and equal to zero) when  $\hat{\theta}_{jk} = 0$  and  $\hat{\sigma}_{jk} = 1$ . But what happens when we plug in the “true” student ability values as before? We have

$$\mathcal{L}_{KL} \Big|_{\hat{\theta}=\theta, \hat{\sigma}=\sigma} = KL(p(\theta|u)||p(\theta)) \quad (2.11)$$

Notice that this is the KL divergence between the **true posterior**  $p(\theta|u)$  and the **true prior**  $p(\theta)$ . This is interpreted as the average difference of number of bits required to encode samples of  $p(\theta|u)$  using a code optimized for  $p(\theta)$ , rather than one optimized for  $p(\theta|u)$ .

### 2.1.3.2 Requirements on sparsity of Q-matrix

TODO: try to show that this is a global minimum for the full VAE loss function. Also take derivatives of the KL loss w.r.t  $\hat{\theta}_{jk}$ . The Q-matrix may help with an identifiability issue (existence of other local minimums) in solving the system  $(a_{ik}\theta_{jk} + b_i)_{jk} = z_i$ . The Q-matrix *may* make the solution unique.

We should be okay with accepting this loss, since the true posterior is not actually known, and we are just using need to look the prior as a reference point into this idea

## 2.2 ML2Pvae Software Package for R

The ML2P-VAE method for parameter estimation has been compiled in an easy-to-use software package for R [4]. This allows researchers who may not have experience with neural networks to implement ML2P-VAE methods on a data set of their choosing. The package **ML2Pvae** is available on the Comprehensive R Archive Network (CRAN) and can be easily installed using the R command `install.packages('ML2Pvae')`

maybe use  
lstinline from  
listings for  
code

### 2.2.1 Package Functionality

**ML2Pvae** uses Tensorflow and Keras to build and train neural networks, but no knowledge of these libraries are required in order to use **ML2Pvae**. The package exports 5 functions available to the user. Two of these are used to construct Keras models, with optional parameters specifying the architecture of the neural network. The only parameters which require input from the user are the number of items on the exam, the number of latent abilities that the exam assesses, and the  $Q$ -matrix relating items and abilities.

The optional inputs in the model construction include a covariance matrix for latent traits, allowing for correlated skills and the implementation described in Section 2.1.1. An important feature for model selection gives the choice of the number of item parameters to use in the logistic IRT model. Though the package is called **ML2Pvae** for the Multidimensional Logistic 2-Parameter model, the package allows for estimating parameters with the 1-Parameter Logistic model, also called the Rasch

model. In this case, there is only a difficulty parameter for each item; each discrimination parameter is fixed to be equal to 1. Other options when building ML2P-VAE models specify the number, size and activation functions of the hidden layers in the encoder.

Using the Keras models returned by the construction functions, **ML2Pvae** provides a function that can be used to train the VAE on data. This function acts as a wrapper for the `fit()` method in the Keras package. The final two methods obtain item parameter estimates and student ability parameter estimates. This is done by grabbing the correct weights/biases from the decoder and feeding student responses through the encoder, respectively.

Should i  
include a short  
code demo?

## CHAPTER 3

### ML2P-VAE RESULTS AND DISCUSSION

The ML2P-VAE method has been used in a various settings in multiple publications [7, 3, 6]. The first paper introduced the method and gives some preliminary results on a small simulated data set. The second, a follow-up presented at the Conference for Artificial Intelligence in Education, displays the advantages that a VAE holds over a regular autoencoder in the task of parameter estimation. The final publication, which has been submitted to Machine Learning, compares different variations of ML2P-VAE with traditional parameter estimation methods on both real and simulated data sets of various sizes.

### 3.1 Description of Data Sets

#### Sim-ECPE

This simulated data set is designed to mirror the real-life Examination for the Certificate of Proficiency in English, detailed further in the next description. Sim-ECPE has 28 items assessing 3 latent traits. Values for the item parameters in the ML2P model were generated from a uniform distribution so that  $a_{ik} \in [0.25, 1.75]$  and  $b_i \in [-3, 3]$ . The range for the discrimination parameters was chosen such that  $0.25 \leq MDISC_i \leq 1.75$  for all  $i$ . Up to 10,000 student abilities  $\Theta \in \mathbb{R}^3$  were sampled from  $\mathcal{N}(0, I)$ . Note that in Sim-ECPE, it is assumed that the latent traits are independent. We use a  $Q$ -matrix consistent with previous literature [8, 18, 12].

Define MDISC



## ECPE

The Examination for the Certificate of Proficiency in English (ECPE) is an exam with 28 items. The set of responses we use is available in the **CDM** package for R [17]. This includes 2,922 students and a  $Q$ -matrix for three skills - “morphosyntactic rules”, “cohesive rules”, and “lexical rules”. Since this is a real-world data set, there are not “true” values of item or student ability parameters to compare with the model estimates.

## Sim-6

A moderately-sized simulated data set, Sim-6 has 50 items evaluating 6 latent traits. The  $Q$ -matrix is also generated randomly, where each entry  $q_{ki}$  is sampled from  $\text{Bern}(0.2)$ . To ensure each item requires at least one latent ability, if a column  $q_{:i} = 0$  after sampling, then one random element in the column is changed to a 1. The discrimination parameters are chosen so that  $a_{ik} \in [0.1, 1.3]$  and  $b_i \in [-3, 3]$ . Abilities  $\Theta \in \mathbb{R}^6$  of 20,000 students were sampled from  $\mathcal{N}(0, \Sigma)$ , where  $\Sigma$  is a correlation matrix with all positive values generated using the SciPy package [21].

## Sim-20

This large data set is generated in a similar manner to Sim-6, but includes 50,000 students, 200 items, and 20 latent traits. The  $Q$ -matrix was generated in the exact same way as that of Sim-6, where  $q_{ki} \sim \text{Bern}(0.2)$ . The difficulty parameters were sampled uniformly so that  $b_i \in [-3, 3]$ , and the discrimination parameters were sampled uniformly so that  $a_{ik} \in [0.1, 0.9]$ . As in Sim-6, the 20 latent abilities are

correlated with one another, and the correlation matrix is generated in the same manner.

## 3.2 Quantitative Results

### 3.2.1 Preliminary Results

IJCNN

### 3.2.2 Ablation Study

This could be helpful

### 3.2.3 Variational Autoencoder vs Autoencoder

Shortly after introducing the ML2P-VAE method, comparisons between a variational autoencoder (VAE) and a regular autoencoder (AE) for parameter estimation were made [3]. Recall that Guo et al. proposed a neural network approach to estimating student mastery in 2017 [10]. This neural network had autoencoding structure, but was geared towards CDM and did not make a connection to IRT or parameter estimation. In this section, we show empirically that using a VAE produces better item and ability estimates than a regular autoencoder and analyze the differences in models leading to this improvement.

AIED

For these experiments, the same simulated data presented in Section 3.2.1 is used here. The neural architecture used for all experiments includes 28 input/output nodes (one for each item), one hidden layer in the encoder with 10 nodes, and an encoded dimension of 3, representing three latent traits. The decoder has no hidden

layers, with connections determined by a given  $Q$ -matrix. Of course, the VAE includes three extra nodes in the encoder output representing variance so that the VAE encoder produces a standard normal distribution.

Model	$a_1$	$a_2$	$a_3$	$b$	Statistic
AE	0.680	0.227	0.529	2.305	AVRB
VAE	0.284	0.159	0.264	1.894	
AE	0.585	0.481	0.534	1.651	RMSE
VAE	0.322	0.346	0.264	1.670	
AE	0.529	0.547	0.748	0.917	CORR
VAE	0.924	0.920	0.986	0.990	

Table 3.1: Statistics for item parameter recovery.

Three error measures for VAE and AE estimates are given in Table 3.1 and Table 3.2. These include absolute value relative bias (AVRB), root mean square error (RMSE) and Pearson correlation (CORR). The statistics for item parameter estimates in Table ??, where  $a_k$  denotes the average measure taken over all items related to latent trait  $\theta_k$ , and  $b$  is the average measure taken over all item difficulty parameters. Note that the AVRB values for difficulty parameters is rather high, likely due to some of the true values of  $b_i$  are very near zero. The item parameter estimates from VAE outperform those from AE for each category and measure. This is corroborated by the correlation plots in Figure 3.1 and Figure 3.2.

Results for student ability parameter estimates are shown in Table 3.2 and Figure 3.3. Again, we see that the error measures from VAE estimates are much

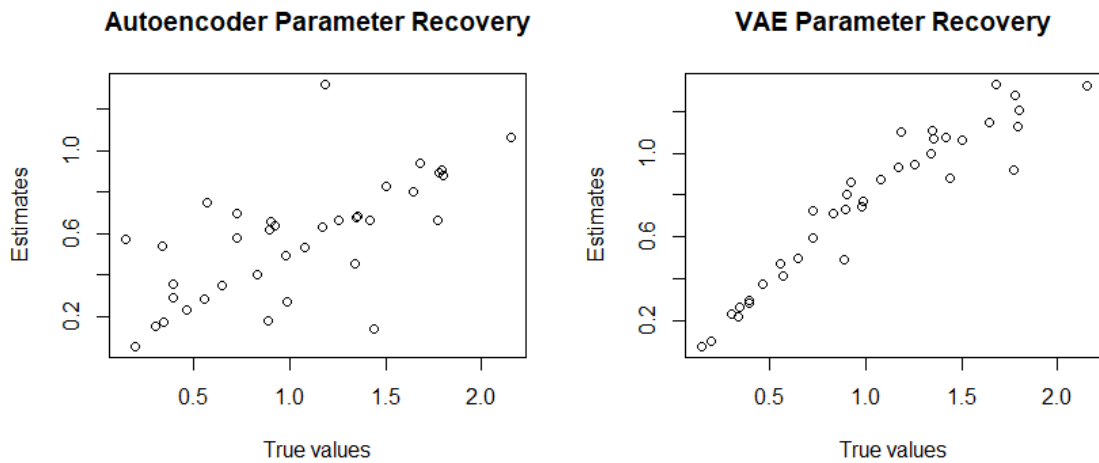


Figure 3.1: Autoencoder and VAE discrimination parameter ( $a_{ji}$ ) recovery.

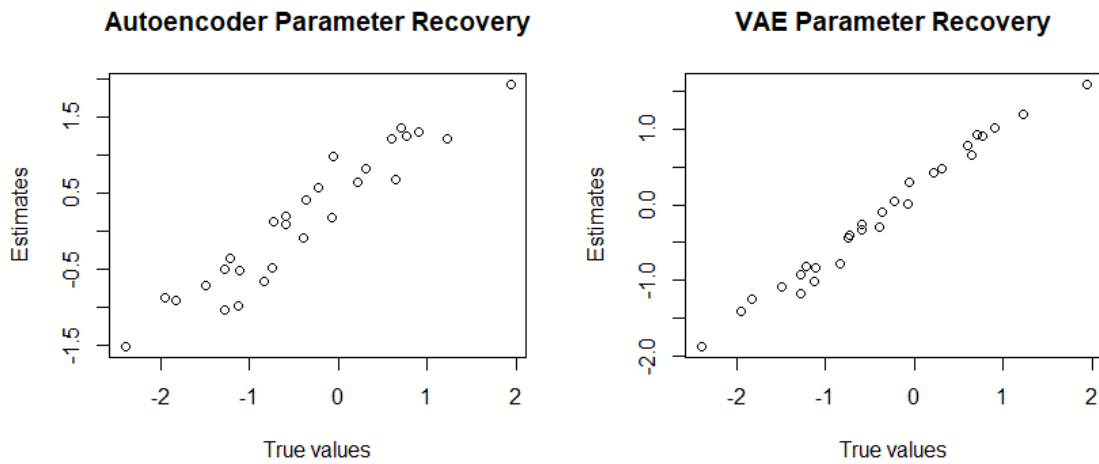


Figure 3.2: Autoencoder and VAE difficulty parameter ( $b_i$ ) recovery.

lower than those from AE. However, the correlation values are slightly better for AE, though the difference is not visible in the correlation plot. The reason that AE has poor error measures yet good correlation is because the ability parameter estimates

are on a different scale than the true values. Notice in the left plot of Figure 3.3 that the vertical axis is on a different scale than that of the right plot. This is likely due to the fact that a VAE has a KL-divergence term in its loss function.

Model	$\theta_1$	$\theta_2$	$\theta_3$	Statistic
AE	7.425	3.107	16.260	AVRB
VAE	1.844	1.713	4.009	
AE	1.788	1.523	1.746	RMSE
VAE	0.664	0.760	0.646	
AE	0.970	0.937	0.971	CORR
VAE	0.965	0.940	0.969	

Table 3.2: Statistics for latent trait prediction.

The lack of a KL-divergence term in an AE also helps explain the poor discrimination parameter estimates shown in the right plot of Figure 3.1. The ML2P model can suffer from an identifiability issue without the assumption that student ability parameters follow some probability distribution [11]. Adding a KL-divergence term in the VAE loss function between the encoder output and the prior  $p(\theta)$ , which is  $\mathcal{N}(0, I)$  in this case.

Both autoencoders and variational autoencoders can be used as IRT parameter estimation methods when a  $Q$ -matrix restricts weights in the decoder. In either case, adding interpretability to neural networks is interesting, but a VAE is able to incorporate an extra piece of domain knowledge in the prior distribution of  $\Theta$ , leading to more accurate estimates.

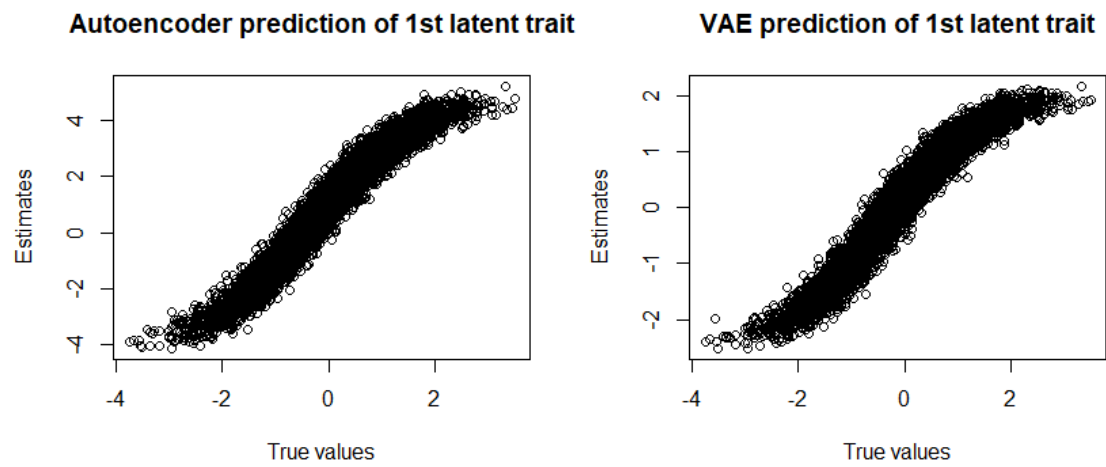


Figure 3.3: Autoencoder and VAE predictions for  $\theta_1$ .

3.2.4 ML2P-VAE vs Traditional Methods

ML

TODO: do 1pl  
in mirt and  
VAE

Data Set	Method	$a$ .RMSE	$a$ .BIAS	$a$ .COR	$b$ .RMSE	$b$ .BIAS	$b$ .COR	$\theta$ .RMSE	$\theta$ .BIAS	$\theta$ .COR	Runtime
(i) 6 abilities Sim-6	MHRM	0.0693	0.0319	0.9986	0.0256	-0.0021	0.9999	0.714	-0.0033	0.7006	1110s
	QMCEM	0.149	-0.067	0.9939	0.0376	-0.002	0.9998	0.7206	0.0023	0.6939	322s
	MCEM	0.1497	-0.0633	0.9936	0.0383	0.0035	0.9997	0.7206	-0.0016	0.6938	1009s
	ML2P-VAE <sub>full</sub>	0.0705	0.0255	0.9985	0.0471	-0.0079	0.9996	0.6649	-0.0178	0.7476	343s
	ML2P-VAE <sub>est</sub>	0.1803	0.0871	0.9891	0.064	-0.0131	0.9993	0.7109	0.0772	0.7082	364s
	ML2P-VAE <sub>ind</sub>	0.1218	-0.0004	0.9944	0.0597	-0.0145	0.9994	0.7222	0.0316	0.6928	252s
(ii) 3 abilities ECPE	MHRM*	0*	0*	1*	0*	0*	1*	0*	0*	1*	162s
	QMCEM	0.0159	0.0035	0.9999	0.0067	-0.0005	1	0.0111	0.0007	0.9999	192s
	MCEM	0.0228	0.0148	0.9998	0.0064	-0.0008	1	0.0132	0.0026	0.9998	33s
	ML2P-VAE <sub>full</sub>	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ML2P-VAE <sub>est</sub>	0.2794	0.2152	0.9713	0.148	0.0951	0.993	0.443	-0.0628	0.8237	61s
	ML2P-VAE <sub>ind</sub>	0.3208	0.2184	0.9504	0.154	0.0872	0.9932	0.3063	0.01	0.9017	49s
(iii) 20 abilities Sim-20	MHRM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	QMCEM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	MCEM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ML2P-VAE <sub>full</sub>	0.078	0.0473	0.9983	0.0608	0.0054	0.9996	0.6145	0.0065	0.7893	1292s
	ML2P-VAE <sub>est</sub>	0.2992	-0.1304	0.9822	0.1655	0.1215	0.9987	0.7364	-0.0276	0.7257	961s
	ML2P-VAE <sub>ind</sub>	0.2043	0.0592	0.9792	0.0958	-0.0029	0.9992	0.7054	0.0747	0.7135	850s

Table 3.3: Error measures for discrimination ( $a$ ), difficulty ( $b$ ), and ability ( $\theta$ ) parameters from various parameter estimation methods on three different data sets. Note that in the ECPE data set, there are no true values, so MHRM estimates are accepted as true. In Sim-20, only ML2P-VAE methods are capable of estimating such high-dimensional latent traits

## CHAPTER 4

### KNOWLEDGE TRACING BACKGROUND

Knowledge Tracing (KT) is a task introduced by Corbett and Anderson in 1995 [?]. Their goal was to model the changing knowledge state of students as they progress through an online intelligent tutoring program. This tutoring system helps students practice writing computer programs by testing them on various rules, such as correct use of in-built functions, and providing feedback on their mistakes. The model tracks each student's knowledge as being in either a learned or unlearned state for each rule. After each interaction, there is a probability  $P(T)$  that a student makes the transition from the unlearned state to the learned state.

The probability that a student has learned a particular rule at timestep  $n$  is

$$P(L_n) = P(L_{n-1}|\text{evidence}) + (1 - P(L_{n-1}|\text{evidence})) \cdot P(T). \quad (4.1)$$

Then the probability of a student performing a task correctly is the sum of the probability that the rule is learned and the student doesn't make a mistake, and the probability that the rule is unlearned but the student guesses correctly.

There are only four parameters for each rule: the probability that the rule is already in the learned state at timestep 0, the probability of transitioning from the unlearned to learned state, the probability of guessing correctly, and the probability of slipping. These parameters are estimated using a hidden Markov Model, and the probability of a the student having learned a rule is updated via Bayes' Theorem.

In recent years, Bayesian Knowledge Tracing (BKT) has been overcome by



deep learning methods. The popularity of neural networks has brought black-box models that yield high accuracy. Many of these methods, detailed in Section 4.2, do not provide a concrete measure of student ability over time. Instead, the only way to track student knowledge is through the predicted probability of them answering questions correctly at a given timestep.

In Chapter 5, new methods using neural networks are presented which produce comparable predictive power to deep learning methods, while providing explainable models with links to Item Response Theory.

## **4.1 Mathematical Setup**

## **4.2 Literature Review**

### **4.2.1 Deep Knowledge Tracing**

\*should explain more because of DKT with Q-matrix

### **4.2.2 Dynamic Key-Value Memory Networks**

\*may not be important to go in-depth

#### **4.2.2.1 Deep-IRT**

Don't need a lot of details here, but they modify DKVMN to gain connection with IRT

### **4.2.3 Self-Attentive Knowledge Tracing**

Important just because this is used in KT-IRT

#### 4.2.4 Performance Factors Analysis

\*should explain more, because of deep-pfa (if that is included)

## CHAPTER 5

### DEEP, INTERPRETABLE METHODS FOR KNOWLEDGE TRACING

#### 5.1 Deep Performance Factors Analysis

#### 5.2 IRT Parameter Recovery + Knowledge Tracing

-AIED 2021 paper

-Tradeoff between accuracy and interpretability

-uses same Q-matrix constraint as ML2P-VAE

## CHAPTER 6

### KNOWLEDGE TRACING RESULTS

#### 6.1 Data Description

Describe each dataset used here.

#### 6.2 Experiment Details

Hyper parameters here

#### 6.3 Results

##### 6.3.1 With Transformers

##### 6.3.2 Deep PFA

##### 6.3.3 IRT parameter recovery + KT

## CHAPTER 7 RELATED WORK

In this chapter, we introduce a few other application areas outside of education where the ML2P-VAE method can be applied. Although the development of the method was done with item response theory in mind, there are other fields which have similar goals to IRT where ML2P-VAE can be applied. As this is not the primary focus of this thesis, we introduce the application setting, draw analogues to education and IRT, and analyze some preliminary results.

### 7.1 Health Sciences

Beck Depression Inventory

-BDI is already connected to IRT -This is interesting because we can add in other features along with response data.

### 7.2 Sports Analytics

Over the past few decades, sports franchises have become more willing to incorporate technology and analytics into their strategy, both on and off the field. For example, the large availability of data has influenced basketball teams to strive for more efficient offense. Off the field, a common task is player evaluation. On sports talk shows, TV personalities often argue over who is the “greatest player of all time”. These arguments are driven by simple in-game measurements as well as more complicated analytics.

Consider  
cutting this  
section down a  
lot  
  
TODO:  
citation.

In 2011, the movie “Moneyball” brought the topic of sports analytics to the public eye, describing the strategy of the 2002 Oakland Athletics baseball team in finding the best valued players. Oakland’s approach was to find players who would reasonably increase the team’s win total, but were undervalued in terms of salary. In general, the task of player evaluation seeks to quantify the contributions of players.

TODO: cite  
moneyball.

### 7.2.1 VAE for Player Evaluation

The ideas of the ML2P-VAE model can be used in this application. Though not all aspects of ML2P-VAE are relevant due to the lack of statistical theory (there is no analogue of IRT models in player evaluation), we can still interpret a hidden layer of the neural network. The goal of this application is to develop new measures for skill quantification of baseball players. This work was originally presented at the Conference on Fuzzy Systems and Data Mining (FSDM) 2019 by Converse et al. [5].

We use a modified VAE for the task of evaluating baseball players in four offensive skill areas. Baseball was chosen over other sports including football and basketball for various reasons: (a) the availability of data, (b) the popularity of analytics in the sport, (c) the consistency of rules and styles of play over the past 50 years, and (d) the lack of positional dependency.

The correspondence to IRT is as follows. Instead of responses to items on an exam, simple measurable statistics (hits, walks, etc.) over the course of a season. The latent ability  $\theta$  in IRT corresponds with the underlying skills required to produce the measurable statistics. In baseball, the four skills chosen are *contact* (how often does

the batter hit the ball), *power* (how hard does the player hit the ball), *baserunning* (is the player good at running the bases), and *pitch intuition* (does the player swing when they are supposed to).

In ML2P-VAE, the core feature which allows for interpretation of the neural network is the  $Q$ -matrix, relating latent abilities with exam items. A similar binary matrix can be constructed associating underlying baseball skills with measurable statistics. For example, the statistic “home runs” requires only the power skill, while avoiding “strikeouts” requires both contact and pitch intuition. A full description of the game statistics used can be found in the appendix.

TODO: add  
 $Q$ -matrix and  
other stuff to  
appendix

This binary matrix determines the non-zero connections between the learned distribution layer (representing baseball skills) and the output layer (reconstructions of the input game statistics). Note that it is reasonable to use a VAE instead of regular autoencoder because it is assumed that among the population of professional baseball players, the distribution of each skill roughly follows a standard normal distribution. This assumption could be altered to allow for correlated underlying skills, but it would be difficult to find an accurate covariance matrix for abstract skills.

### 7.2.2 Experiments

Data was gathered from Major League Baseball players from 1950-2018, yielding 8,604 samples, where each data point corresponds to a particular player’s performance in a particular year. 13 measurable game statistics were chosen as inputs to the VAE: singles (1B), doubles (2B), home runs (HR), runs (R), runs batted in (RBI),

Should add  
citation to  
data

walks (BB), intentional walks (IBB), strikeouts (K), sacrifice (SAC), grounded into double play (GDP), stolen bases (SB), caught stealing (CS), and walk/strikeout ratio (BB/K). Each statistic was rescaled using Gaussian normalization so that each input feature was centered at 0 with variance one. Additionally, the game statistics strikeouts and caught stealing were multiplied by  $-1$  for each observation so that a larger number is more desirable. As in the ML2P-VAE architecture, the decoder weights are constrained to be non-negative, so that a higher skill value can only increase the reconstructed in-game statistics.

Since this is an unsupervised method with the goal of creating *new* skill measures, it is difficult to evaluate the obtained results. Instead, we take the more commonly used baseball statistics contact rate (CR), speed score (SPD), isolated power (ISO), and on-base percentage (OBP) to compare with the skills *contact*, *baserunning*, *power*, and *pitch intuition*, respectively. More information on how these measures are calculated can be found in the appendix.

TODO: add to  
appendix

Each skill is plotted against its evaluation statistic in Figure 7.1 [5]. Note that each of the four plots display high correlation, but do not match exactly. This is desirable in the sense that our new skill quantifications do in fact measure what they are intended to measure, but give new insights and ranking for each player. Though these results could be improved, it is clear that variations of the ML2P-VAE model can be applied in areas other than education.



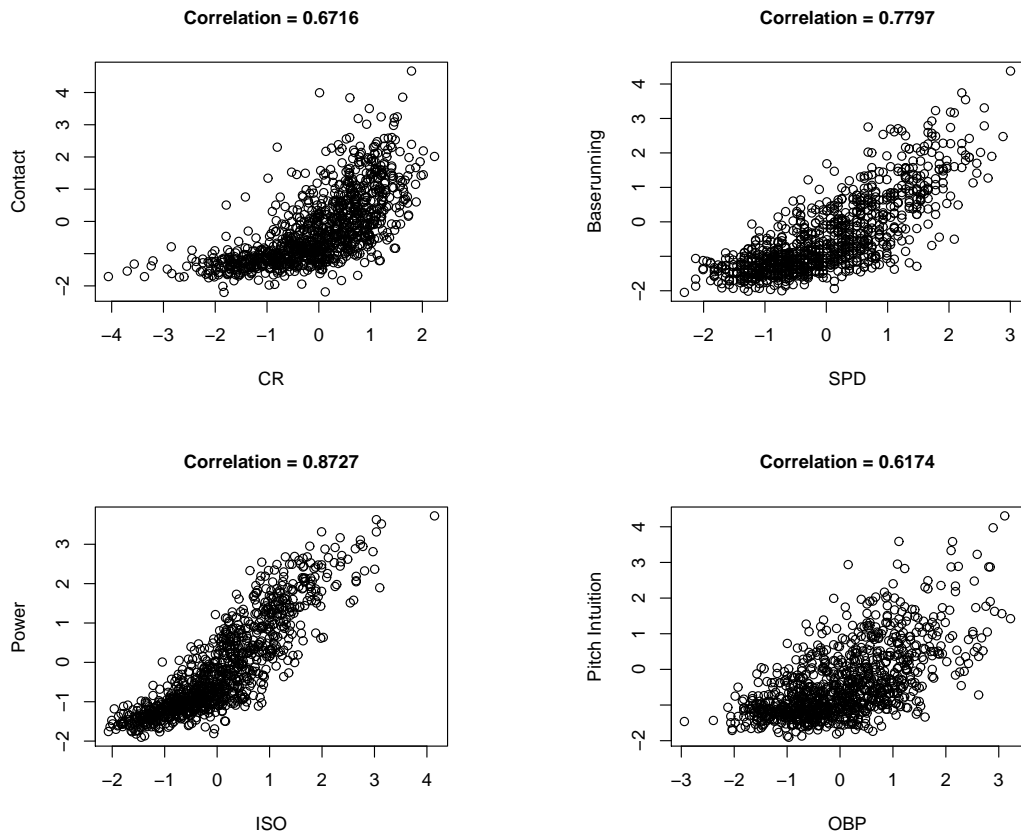


Figure 7.1: Each latent skill's estimates plotted against its evaluation statistic.

## REFERENCES

- [1] F. Baker and S. Kim. *Item Response Theory Parameter Estimation Techniques*. Taylor & Francis Group, 2nd edition, 2004.
- [2] D.M. Blei, A. Kucukelbir, and J.D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.
- [3] G. Converse, M. Curi, and S. Oliveira. Autoencoders for educational assessment. In *International Conference on Artificial Intelligence in Education (AIED)*, 2019.
- [4] Geoffrey Converse. *ML2Pvae: Variational Autoencoder Models for IRT Parameter Estimation*, 2020. R package version 1.0.0.
- [5] Geoffrey Converse, Brooke Arnold, Mariana Curi, and Suely Oliveira. Variational autoencoders for baseball player evaluation. In Antonio J. Tallón-Ballesteros, editor, *Fuzzy Systems and Data Mining V - Proceedings of FSDM 2019, Kitakyushu City, Japan, October 18-21, 2019*, volume 320 of *Frontiers in Artificial Intelligence and Applications*, pages 305–311. IOS Press, 2019.
- [6] Geoffrey Converse, Mariana Curi, Suely Oliveira, and Jonathan Templin. Estimation of multidimensional item response theory models with correlated latent variables using variational autoencoders. *Machine Learning*, Under review 2020.
- [7] M. Curi, G. Converse, J. Hajewski, and S. Oliveira. Interpretable variational autoencoders for cognitive models. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.
- [8] M.A. da Silva, R. Liu, A.C. Huggins-Manley, and J.L. Bazan. Incorporating the q-matrix into multidimensional item response theory models. *Educational and Psychological Measurement*, 2018.
- [9] Carl Doersch. Tutorial on variational autoencoders, 2016.
- [10] Q. Guo, M. Cutumisu, and Y. Cui. A neural network approach to estimate student skill mastery in cognitive diagnostic assessments. In *10th International Conference on Educational Data Mining*, 2017.
- [11] Shelby J. Haberman. Identifiability of parameters in item response models with unconstrained ability distributions. Technical Report RR-05-24, Research and Development, ETS, December 2005.

- [12] R. Henson and J. Templin. Large-scale language assessment using cognitive diagnosis models. In *Annual meeting of the National Council for Measurement in Education*, 2007.
- [13] D. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.
- [14] F. Lord and M. R Novick. *Statistical theories of mental test scores*. IAP, 1968.
- [15] Elad Plaut. From principal subspaces to principal components with linear autoencoders. *arXiv:1804.10253v2*, 2018.
- [16] Hassan Rafique, Tong Wang, Quihang Lin, and Arshia Sighani. Transparency promotion with model-agnostic linear competitors. In *Proceedings of the International Conference on Machine Learning*, 2020.
- [17] Alexander Robitzsch, Thomas Kiefer, Ann Cathrice George, and Ali Uenlue. *CDM: Cognitive Diagnosis Modeling*, 2020. R package version 7.5-15.
- [18] Jonathan Templin and Lesa Hoffman. Obtaining diagnostic classification model estimates using mplus. *Educational Measurement: Issues and Practice*, 32(2):37–50, 2013.
- [19] David Thissen and Howard Wainer. *Test Scoring*. Lawrence Erlbaum Associates Publishers, 2001.
- [20] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pages 1096–1103, 2008.
- [21] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stefan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, Ihan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antonio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 2020.
- [22] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018.