Neural Network Methods for Application in Educational Measurement

Geoffrey Converse

University of Iowa

July 15, 2021

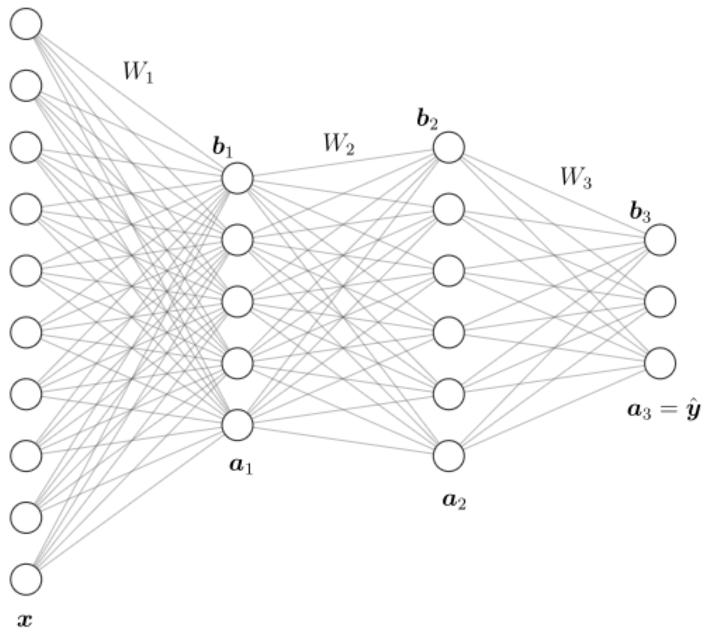
PhD Defense in Applied Mathematical and Computational Sciences

Overview

- How can we quantify student learning?
- How can we deal with large datasets?

Outline

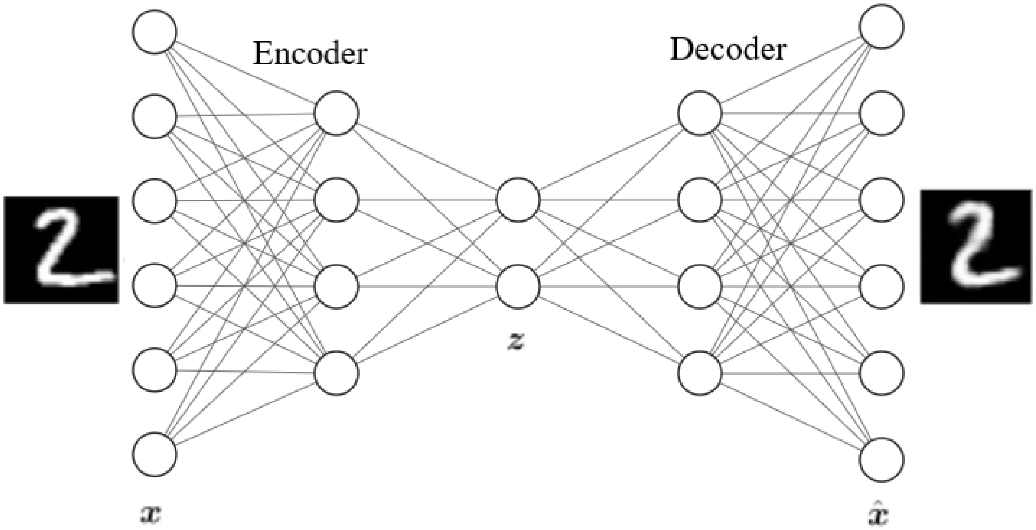
Artificial Neural Networks (ANN)



Input $\boldsymbol{x}$, approximate a true target $\boldsymbol{y}$ via a series of (learned) linear transformations W_l and nonlinear re-scaling $\sigma(\cdot) : \mathbb{R}^d \to (0,1)^d$

$$\hat{\boldsymbol{y}} = \sigma(W_3\sigma(W_2\sigma(W_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3)$$

Autoencoder (AE)



- Encode data into smaller dimension
  - Image compression
  - Non-linear PCA
- Reconstruct original input by minimizing $\mathcal{L} = ||\boldsymbol{x} - \hat{\boldsymbol{x}}||$

Variational Autoencoder (VAE)

- Observed data $\boldsymbol{x}$ is generated by some latent code $\boldsymbol{z}$
- Latent code is assumed to follow a normal distribution $p_z^*(\boldsymbol{z}) = \mathcal{N}(0, I)$
- If $\boldsymbol{z}$ is high dimensional, the posterior is intractable:

$$p_z^*(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_x^*(\boldsymbol{x}|\boldsymbol{z})p_z^*(\boldsymbol{z})}{\int p_x^*(\boldsymbol{x}|\boldsymbol{z})p_z^*(\boldsymbol{z})\,d\boldsymbol{z}}$$

- Approximate the true posteriors $p_z^*(\boldsymbol{z}|\boldsymbol{x})$ and $p_x^*(\boldsymbol{x}|\boldsymbol{z})$ with neural networks $q_\alpha(\boldsymbol{z}|\boldsymbol{x})$ and $p_\beta(\boldsymbol{x}|\boldsymbol{z})$

VAE Derivation

$$\begin{aligned}
\log p_x^*(\boldsymbol{x}) &= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log p_x^*(\boldsymbol{x}) \, d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p_z^*(\boldsymbol{z}|\boldsymbol{x}) p_x^*(\boldsymbol{x})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p^*(\boldsymbol{x}, \boldsymbol{z})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \left(\log \frac{q_\alpha(\boldsymbol{z}|\boldsymbol{x})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} + \log \frac{p^*(\boldsymbol{x}, \boldsymbol{z})}{q_\alpha(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \mathcal{D}_{KL} \left[q_\alpha(\cdot|\boldsymbol{x}) || p_z^*(\cdot|\boldsymbol{x}) \right] + \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p^*(\boldsymbol{x}, \boldsymbol{z})}{q_\alpha(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \mathcal{D}_{KL} \left[q_\alpha(\cdot|\boldsymbol{x}) || p_z^*(\cdot|\boldsymbol{x}) \right] + \mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})} \left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p^*(\boldsymbol{x}, \boldsymbol{z}) \right] \\
&= \mathcal{D}_{KL} \left[q_\alpha(\cdot|\boldsymbol{x}) || p_z^*(\cdot|\boldsymbol{x}) \right] + \tilde{\mathcal{L}}_*(\alpha; \boldsymbol{x})
\end{aligned}$$

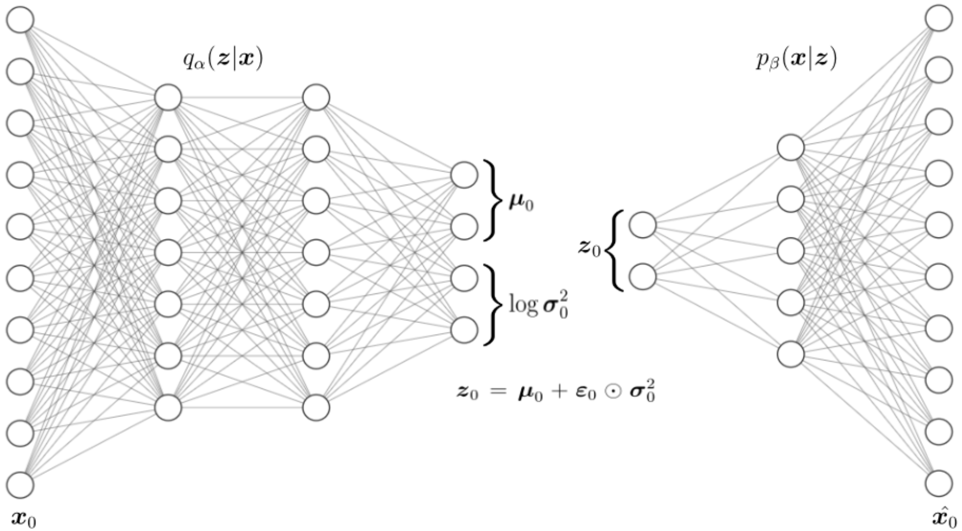
VAE Derivation

- KL-Divergence is non-negative, so we look at the evidence lower bound (ELBO) $\tilde{\mathcal{L}}_*$

$$\begin{aligned}\log p_x^*(\boldsymbol{x}) \geq \tilde{\mathcal{L}}_*(\alpha; \boldsymbol{x}) &= \mathbb{E}_{z \sim q_\alpha(\cdot|\boldsymbol{x})}\left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p^*(\boldsymbol{x}, \boldsymbol{z})\right] \\ &= \mathbb{E}_{z \sim q_\alpha(\cdot|\boldsymbol{x})}\left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p_x^*(\boldsymbol{x}|\boldsymbol{z}) + \log p_z^*(\boldsymbol{z})\right] \\ &\approx \mathbb{E}_{z \sim q_\alpha(\cdot|\boldsymbol{x})}\left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p_\beta(\boldsymbol{x}|\boldsymbol{z}) + \log p_z^*(\boldsymbol{z})\right] \\ &= -\mathcal{D}_{KL}\left[q_\alpha(\cdot|\boldsymbol{x}) \middle|\middle| p_z^*(\cdot)\right] + \mathbb{E}_{z \sim q_\alpha(\cdot|\boldsymbol{x})}\left[\log p_\beta(\boldsymbol{x}|\boldsymbol{z})\right] \\ &= \tilde{\mathcal{L}}(\alpha, \beta; \boldsymbol{x})\end{aligned}$$

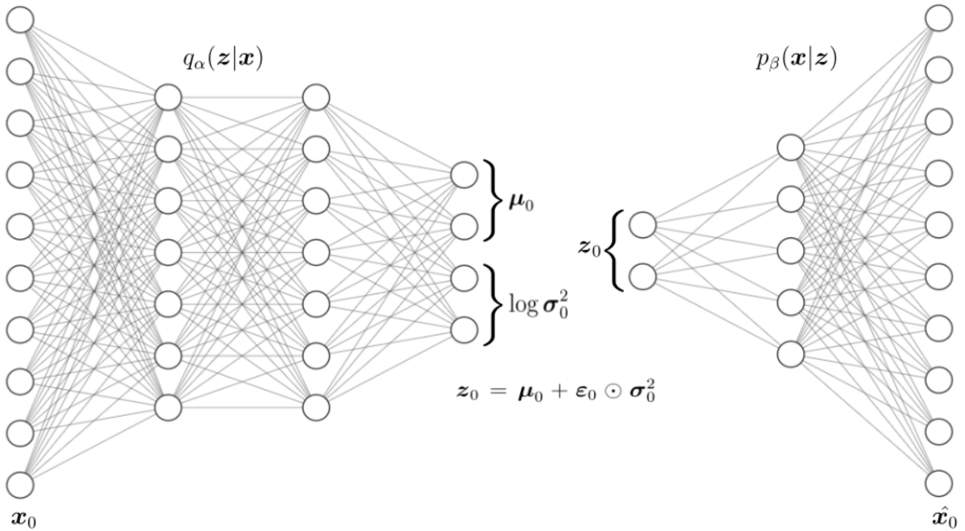
- We've replaced all unknown distributions $p_\cdot^*(\cdot)$ with assumed or approximate distributions
- VAE loss is given as $\mathcal{L}(\alpha, \beta; \boldsymbol{x}) = -\tilde{\mathcal{L}}(\alpha, \beta; \boldsymbol{x})$ where α and β reference the trainable parameters in the encoder and decoder

Variational Autoencoder (VAE)



- Fit encoded space to $\boldsymbol{z} \sim \mathcal{N}(0, I)$
- Given input $\boldsymbol{x}_0$, the encoder outputs a distribution $q_\alpha(\boldsymbol{z}|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2)$

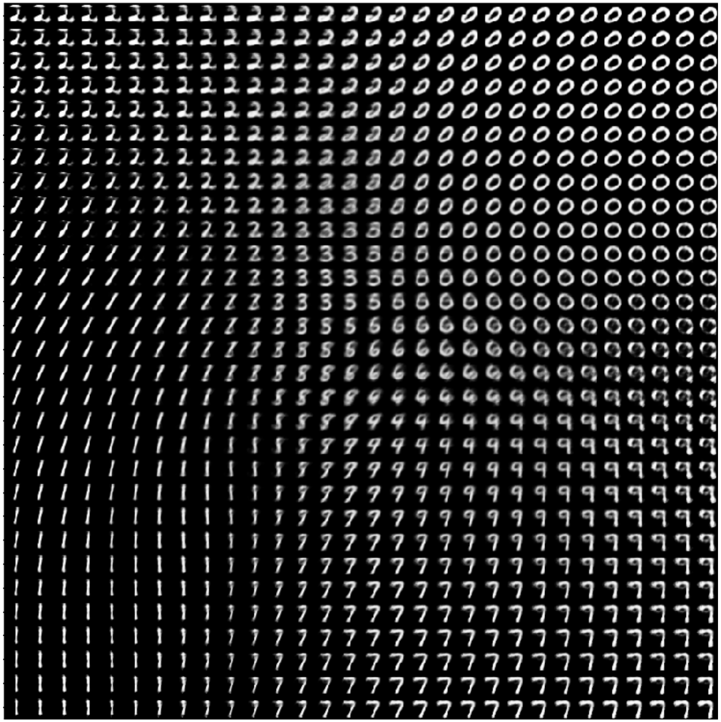
Variational Autoencoder (VAE)



- Fit encoded space to $\boldsymbol{z} \sim \mathcal{N}(0, I)$
- Given input $\boldsymbol{x}_0$, the encoder outputs a distribution $q_\alpha(\boldsymbol{z}|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2)$
- Sample $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$, set $\boldsymbol{z}_0 = \boldsymbol{\mu}_0 + \boldsymbol{\varepsilon} \odot \boldsymbol{\sigma}_0$
- Feed $\boldsymbol{z}_0$ through decoder to obtain reconstruction $\hat{\boldsymbol{x}}_0 \sim p_\beta(\cdot|\boldsymbol{z}_0)$

Variational Autoencoder (VAE)

- VAE are used as a generative model
- Train on a set of images, then generate *new* images which are similar to the training data by sampling form the latent space



Item Response Theory (IRT)

- Goal: Explain relationship between student ability and exam performance
- Each student has a latent "ability" value $\theta \in \mathbb{R}$

Item Response Theory (IRT)

- Goal: Explain relationship between student ability and exam performance
- Each student has a latent "ability" value $\theta \in \mathbb{R}$
  - θ is not directly observable
  - Naive solution: $\theta \approx \dfrac{\text{questions answered correctly}}{\text{total number of questions}}$

Item Response Theory (IRT)

- Goal: Explain relationship between student ability and exam performance
- Each student has a latent "ability" value $\theta \in \mathbb{R}$
  - θ is not directly observable
  - Naive solution: $\theta \approx \dfrac{\text{questions answered correctly}}{\text{total number of questions}}$
- For an assessment with n items taken by N subjects, what is the probability that student j answers item i correctly?

$$P(u_{ij} = 1 | \theta_j) = f(\theta_j; \Lambda_i)$$

- θ_j = latent ability of subject j
- Λ_i = set of parameters for item i (e.g. difficulty)

Rasch Model

- Define $\delta_i > 0$ as the difficulty of item i, and $\eta_j > 0$ the ability of subject j.
- Rasch: Probability of success depends on ratio $\dfrac{\delta_i}{\eta_j}$

$$P(u_{ij} = 1|\eta_j, \delta_i) = \frac{1}{1 + \delta_i/\eta_j} = \frac{\eta_j}{\eta_j + \delta_i}$$

Rasch Model

- Define $\delta_i > 0$ as the difficulty of item i, and $\eta_j > 0$ the ability of subject j.
- Rasch: Probability of success depends on ratio $\dfrac{\delta_i}{\eta_j}$

$$P(u_{ij} = 1|\eta_j, \delta_i) = \frac{1}{1 + \delta_i/\eta_j} = \frac{\eta_j}{\eta_j + \delta_i}$$

- Logarithmic transformation: $\theta_j = \log \eta_j$ and $\beta_i = \log \delta_i$
- Rasch Model:

$$P(u_{ij} = 1|\theta_j, \beta_i) = \frac{1}{1 + e^{\beta_i - \theta_j}}$$

2-Parameter Logistic Model (2PL)

- Probability of a correct response follows the logistic equation:

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

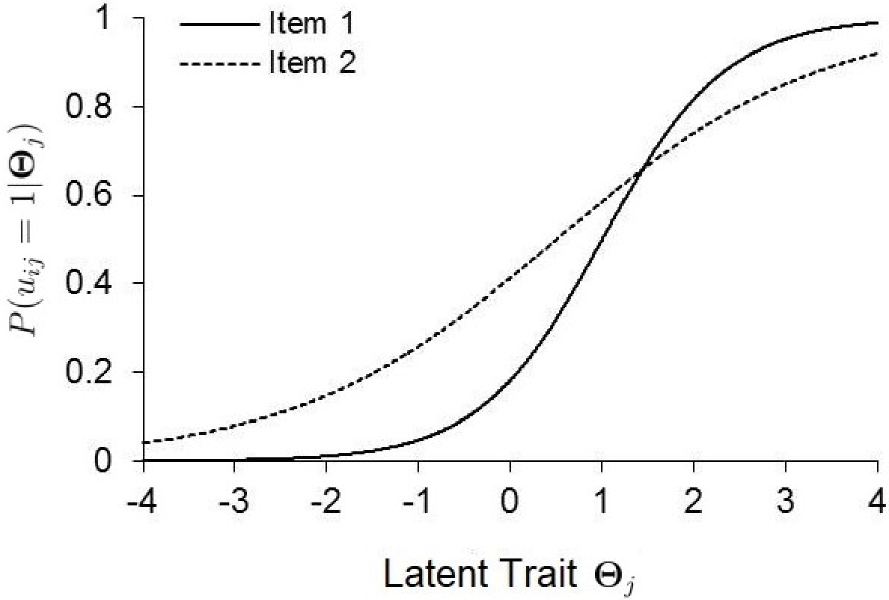
2-Parameter Logistic Model (2PL)

- Probability of a correct response follows the logistic equation:

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

- a_i = discrimination parameter (slope)
  - Quantifies the capability of item i in differentiating between students with sufficient/insufficient ability
- b_i = difficulty parameter (intercept)

Item Characteristic Curve (ICC)



- Item 1 has higher discrimination than Item 2

Multidimensional IRT

- Now assume that an assessment is testing K skills
  - For example, a math exam can test skills add, subtract, multiply, divide
  - Student j has a vector of skills $\Theta_j = (\theta_{j1}, .., \theta_{jK})^T$
  - Multiple skills can be assessed by a single item

Multidimensional IRT

- Now assume that an assessment is testing K skills
  - For example, a math exam can test skills add, subtract, multiply, divide
  - Student j has a vector of skills $\Theta_j = (\theta_{j1}, .., \theta_{jK})^T$
  - Multiple skills can be assessed by a single item
- Binary Q-matrix defines relationship between items and skills
  - $Q \in \mathbb{R}^{n \times K}$,

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

Multidimensional Logistic 2-Parameter (ML2P) Model

- Probability of correct response given by:

$$
\begin{aligned}
P(u_{ij} = 1 | \Theta_j; \boldsymbol{a}_i, b_i) &= \frac{1}{1 + \exp\left[-\boldsymbol{a}_i^\top \Theta_j + b_i\right]} \\
&= \frac{1}{1 + \exp[-\sum_{k=1}^{K} q_{ik} a_{ik} \theta_{jk} + b_i]}
\end{aligned}
$$

Multidimensional Logistic 2-Parameter (ML2P) Model

- Probability of correct response given by:

$$P(u_{ij} = 1 | \Theta_j; \boldsymbol{a}_i, b_i) = \frac{1}{1 + \exp\left[-\boldsymbol{a}_i^\top \Theta_j + b_i\right]} = \frac{1}{1 + \exp[-\sum_{k=1}^{K} q_{ik} a_{ik} \theta_{jk} + b_i]}$$

  - a_{ik} = discrimination parameter between item i and skill k
  - b_i = difficulty parameter

Estimating IRT Parameters

- In application, given only binary matrix of N response sets $U \in \mathbb{R}^{N \times n}$
  - $\boldsymbol{u}_j \in \mathbb{R}^n$ details student j's correct/incorrect responses to n items
- How to obtain the item parameters $\boldsymbol{a}_i$ and b_i and student ability parameters $\boldsymbol{\Theta}_j$?
- Maximize the log-likelihood of the data

$$\log L = \sum_{j=1}^{N} \sum_{i=1}^{n} u_{ij} \log P(u_{ij} = 1) + (1 - u_{ij}) \log P(u_{ij} = 0)$$

Joint Maximum Likelihood Estimation (JMLE)

- Estimate student and item parameters simultaneously
- Gradient vector $\boldsymbol{f}(\boldsymbol{x}) = \nabla_{\theta,a,b} \log L\big|_{\boldsymbol{x}}$
- Jacobian $J(\boldsymbol{x}) = \left[\dfrac{\partial^2 \log L}{\partial x \partial y}\right]\Bigg|_{\boldsymbol{x}} \in \mathbb{R}^{(NK+nK+n)\times(NK+nK+n)}$
  - $x, y \in \{\theta_{jk}, a_{ik}, b_i\}_{j,k,i}$
- Newton-Raphson iterations

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - J^{-1}(\boldsymbol{x}_t)\boldsymbol{f}(\boldsymbol{x}_t)$$

Joint Maximum Likelihood Estimation (JMLE)

- Estimate student and item parameters simultaneously
- Gradient vector $\boldsymbol{f}(\boldsymbol{x}) = \nabla_{\theta,a,b} \log L\big|_{\boldsymbol{x}}$
- Jacobian $J(\boldsymbol{x}) = \left[\dfrac{\partial^2 \log L}{\partial x \partial y}\right]\Bigg|_{\boldsymbol{x}} \in \mathbb{R}^{(NK+nK+n)\times(NK+nK+n)}$
  - $x, y \in \{\theta_{jk}, a_{ik}, b_i\}_{j,k,i}$
- Newton-Raphson iterations

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - J^{-1}(\boldsymbol{x}_t)\boldsymbol{f}(\boldsymbol{x}_t)$$

- J can be very large, difficult to invert
- Possibly unbounded parameter estimates

Marginal Maximum Likelihood (MMLE)

TODO: clean and summarize MMLE in one slide

- Assume that $\boldsymbol{\Theta}$ follows some distribution $g(\boldsymbol{\Theta})$
- Maximize the mariginal likelihood

$$L = \prod_{j=1}^{N} P(\boldsymbol{u}_j) = \prod_{j=1}^{N} \int P(\boldsymbol{u}_j|\boldsymbol{\Theta})\, g(\boldsymbol{\Theta})\, d\boldsymbol{\Theta}$$

Marginal Maximum Likelihood (MMLE)

TODO: clean and summarize MMLE in one slide

- Assume that $\boldsymbol{\Theta}$ follows some distribution $g(\boldsymbol{\Theta})$
- Maximize the mariginal likelihood

$$L = \prod_{j=1}^{N} P(\boldsymbol{u}_j) = \prod_{j=1}^{N} \int P(\boldsymbol{u}_j|\boldsymbol{\Theta})\, g(\boldsymbol{\Theta})\, d\boldsymbol{\Theta}$$

- The EM algorithm:
  - Compute expectation of $\boldsymbol{\Theta}$
    - Compute K dimensional integral
  - Maximize L with respect to item parameters

Difficulties of IRT Parameter Estimation

TODO: summarize the problems with high-dim theta

- High dimensional IRT is hard
- Large matrix inversion
- High-dimensional integral

Similarities between IRT and VAE

TODO: I think there is one more thing

- IRT and VAE assume normally distributed latent space
  - Observed data is generated from latent code

Similarities between IRT and VAE

TODO: I think there is one more thing

- IRT and VAE assume normally distributed latent space
  - Observed data is generated from latent code
- ML2P model and sigmoidal activation function:

$$P(u_{ij} = 1 | \mathbf{\Theta}_j) = \frac{1}{1 + \exp[-\sum_{k=1}^{K} a_{ik}\theta_{jk} + b_i]}$$

$$\sigma(z) = \sigma(\vec{w}^T\vec{a} + b) = \frac{1}{1 + \exp[-\sum_{k=1} w_k a_k - b]}$$

Model Description

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$

Model Description

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$
- No hidden layers in the VAE decoder
- Restrict nonzero weights in the decoder according to Q-matrix
- Require decoder weights to be nonnegative
  - Avoids reflection $\theta \cdot (-a) = (-\theta) \cdot a$

Model Description

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$
- No hidden layers in the VAE decoder
- Restrict nonzero weights in the decoder according to Q-matrix
- Require decoder weights to be nonnegative
  - Avoids reflection $\theta \cdot (-a) = (-\theta) \cdot a$

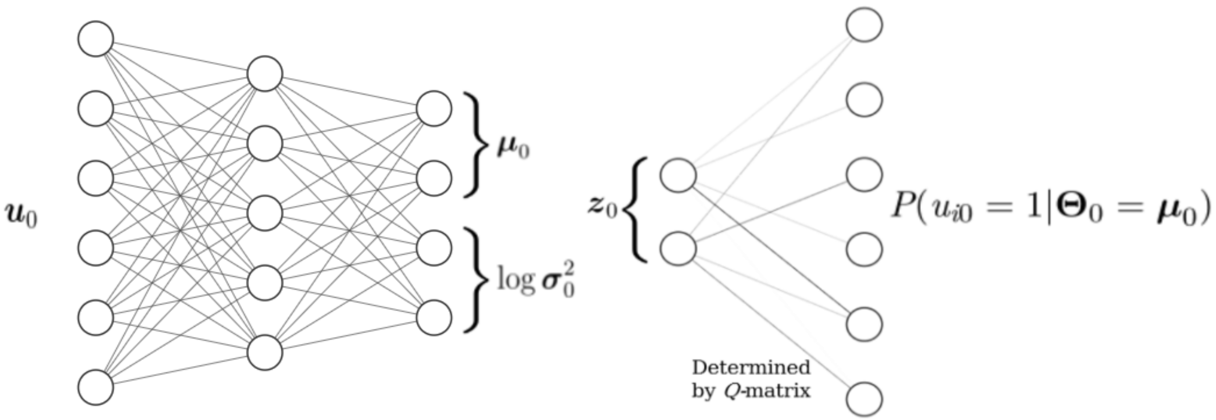
Model Description

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$
- No hidden layers in the VAE decoder
- Restrict nonzero weights in the decoder according to Q-matrix
- Require decoder weights to be nonnegative
  - Avoids reflection $\theta \cdot (-a) = (-\theta) \cdot a$
- Sigmoidal activation function in output layer

Model Description

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$
- No hidden layers in the VAE decoder
- Restrict nonzero weights in the decoder according to Q-matrix
- Require decoder weights to be nonnegative
  - Avoids reflection $\theta \cdot (-a) = (-\theta) \cdot a$
- Sigmoidal activation function in output layer
- Decoder interpreted as the ML2P model
  - Activation of nodes in encoded layer $\Rightarrow$ latent ability estimates
  - Weights in decoder $\Rightarrow$ discrimination parameter estimates
  - Bias of output nodes $\Rightarrow$ difficulty parameter estimates
  - Output layer $\Rightarrow$ probability of answering items correctly

ML2P-VAE



- Trainable weights in decoder are item parameter estimates
- Feed responses $\boldsymbol{u}_0$ through encoder to obtain ability estimates $\boldsymbol{\Theta}_0 = \boldsymbol{\mu}_0$

Advantages of ML2P-VAE Approach

For the IRT application:

- No trouble for high-dimensional $\boldsymbol{\Theta}$
- Doesn't directly optimize $\boldsymbol{\Theta}$
  - Large number of students isn't a computational burden
- Learning a *function* that maps responses to latent abilities
  - Encoder: $\boldsymbol{u}_0 \mapsto \boldsymbol{\Theta}_0$

Advantages of ML2P-VAE Approach

For the IRT application:

- No trouble for high-dimensional $\boldsymbol{\Theta}$
- Doesn't directly optimize $\boldsymbol{\Theta}$
    - Large number of students isn't a computational burden
- Learning a *function* that maps responses to latent abilities
    - Encoder: $\boldsymbol{u}_0 \mapsto \boldsymbol{\Theta}_0$

In the machine learning field:

- Ability to interpret a hidden neural layer
- Less abstract encoded latent space
- Explainable trainable parameters in decoder

Advantages of ML2P-VAE Approach

For the IRT application:

- No trouble for high-dimensional $\boldsymbol{\Theta}$
- Doesn't directly optimize $\boldsymbol{\Theta}$
  - Large number of students isn't a computational burden
- Learning a *function* that maps responses to latent abilities
  - Encoder: $\boldsymbol{u}_0 \mapsto \boldsymbol{\Theta}_0$

In the machine learning field:

- Ability to interpret a hidden neural layer
- Less abstract encoded latent space
- Explainable trainable parameters in decoder

Method originally presented at the International Joint Conference on Neural Networks (IJCNN) 2019

VAE vs AE Comparison

TODO: clean up and be better

- Guo, Cutumisu, and Cui proposed using AE in skill estimation
- Directly compare neural networks in ML2P application
  - Item parameter recovery
  - Skill estimation

VAE vs AE Comparison

TODO: clean up and be better

- Guo, Cutumisu, and Cui proposed using AE in skill estimation
- Directly compare neural networks in ML2P application
  - Item parameter recovery
  - Skill estimation
- How useful is prior $p(\boldsymbol{\Theta})$?
- What is the effect of the KL-Divergence term in the loss function?

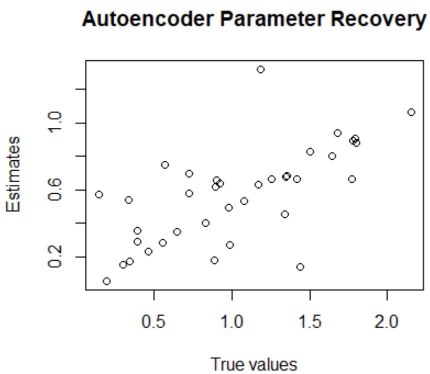
VAE vs AE Comparison

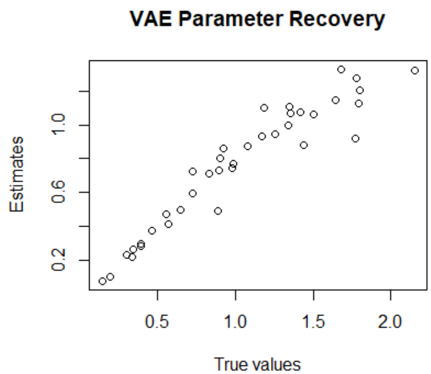
TODO: clean up and be better

- Guo, Cutumisu, and Cui proposed using AE in skill estimation
- Directly compare neural networks in ML2P application
  - Item parameter recovery
  - Skill estimation
- How useful is prior $p(\boldsymbol{\Theta})$?
- What is the effect of the KL-Divergence term in the loss function?

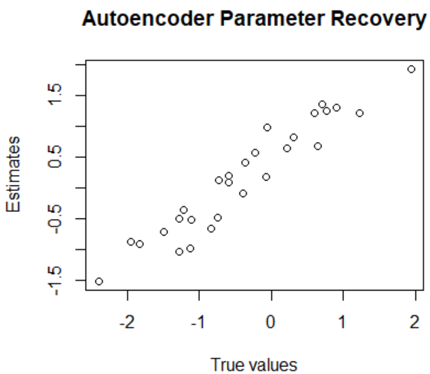
Results presented at Artificial Intelligence in Education (AIED) 2019

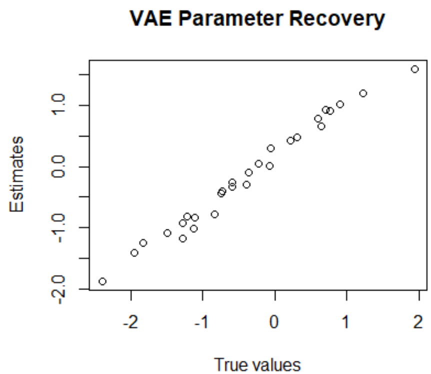
VAE vs AE Comparison





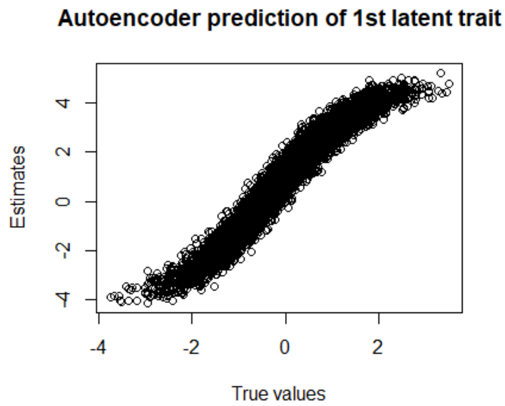
Discrimination parameters a_{ik}

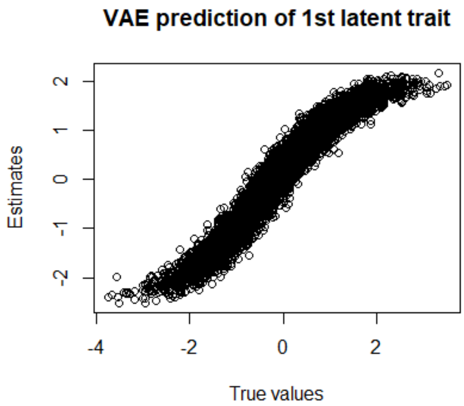




Difficulty parameters b_i

VAE vs AE Comparison





- Similar skill estimate correlation, but on different scale
- VAE much more accurate parameter recovery

Correlated Latent Traits in IRT

- In real applications, independent skills are not realistic: $\mathbf{\Theta} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, not $\mathcal{N}(0, I)$.
  - Example: students who are good at addition are also good at subtraction

Correlated Latent Traits in IRT

- In real applications, independent skills are not realistic: $\mathbf{\Theta} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, not $\mathcal{N}(0, I)$.
  - Example: students who are good at addition are also good at subtraction
- Covariance matrix is symmetric, positive definite matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & c_{12} & \cdots & c_{1k} \\ c_{21} & \sigma_2^2 & \cdots & c_{2k} \\ \vdots & & \ddots & \vdots \\ c_{k1} & \cdots & c_{k(k-1)} & \sigma_k^2 \end{bmatrix}$$

- With variances σ_i^2 and covariances $c_{ij} = c_{ji}$

Correlated Latent Code in VAE

- In most VAE applications, it is convenient to assume latent code $\boldsymbol{z}$ is *independent*
  - Forces each dimension of $\boldsymbol{z}$ to measure different features
  - $\boldsymbol{z}$ is *abstract*, with **no real-world understanding**

Correlated Latent Code in VAE

- In most VAE applications, it is convenient to assume latent code $\boldsymbol{z}$ is *independent*
  - Forces each dimension of $\boldsymbol{z}$ to measure different features
  - $\boldsymbol{z}$ is *abstract*, with **no real-world understanding**
- For ML2P-VAE, we know that latent code $\boldsymbol{z}$ approximates latent traits $\boldsymbol{\Theta}$
  - We may have **domain knowledge** of the distribution of $\boldsymbol{\Theta}$

KL-Divergence for Multivariate Gaussians

- KL-Divergence between two K-dimensional multivariate Gaussian distributions:

$$\mathcal{D}_{KL}\left[\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0) || \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)\right] =$$

$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \Sigma_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - K + \ln\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right)$$

KL-Divergence for Multivariate Gaussians

- KL-Divergence between two K-dimensional multivariate Gaussian distributions:

$$\mathcal{D}_{KL}\left[\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0) || \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)\right] =$$
$$\frac{1}{2}\left(\text{tr}(\Sigma_1^{-1}\Sigma_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \Sigma_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - K + \ln\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right)$$

- When fitting a VAE, $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ is assumed to be known, so $\boldsymbol{\mu}_1$ and Σ_1 are constant
- $\boldsymbol{\mu}_0$ and Σ_0 obtained from feeding one sample through the encoder

Implementation Requirements for Correlated VAE

1. KL Divergence calculation uses $\boldsymbol{\mu}_0$, Σ_0, and $\ln \det \Sigma_0$
2. Sample from a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$:

Implementation Requirements for Correlated VAE

1. KL Divergence calculation uses $\boldsymbol{\mu}_0$, Σ_0, and $\ln\det\Sigma_0$
   - Require $\det\Sigma_0 > 0$ for <u>any</u> input $\boldsymbol{u}_0$
   - Σ_0 is a function of the input $\boldsymbol{u}_0$ and every encoder weight
2. Sample from a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$:

Implementation Requirements for Correlated VAE

1. KL Divergence calculation uses $\boldsymbol{\mu}_0$, Σ_0, and $\ln\det\Sigma_0$
   - Require $\det\Sigma_0 > 0$ for <u>any</u> input $\boldsymbol{u}_0$
   - Σ_0 is a function of the input $\boldsymbol{u}_0$ and every encoder weight
2. Sample from a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$:
   - Find a matrix G such that $GG^T = \Sigma_0$
   - Sample $\boldsymbol{\varepsilon} = (\varepsilon_1, ..., \varepsilon_k)^T$ with each $\varepsilon_i \sim \mathcal{N}(0, 1)$
   - Generate sample $\boldsymbol{z}_0 = \boldsymbol{\mu}_0 + G\boldsymbol{\varepsilon}$

Correlated VAE Implementation

- Architecture: Encoder outputs $K + K(K+1)/2$ nodes
  - K nodes for $\boldsymbol{\mu}_0$, and $K(K+1)/2$ nodes for L_0 lower triangular

Correlated VAE Implementation

- Architecture: Encoder outputs $K + K(K+1)/2$ nodes
  - K nodes for $\boldsymbol{\mu}_0$, and $K(K+1)/2$ nodes for L_0 lower triangular
- Sampling: Calculate $G_0 = e^{L_0}$
  - Note G_0 is lower triangular, nonsingular
  - Send sample $\boldsymbol{z} = \boldsymbol{\mu}_0 + G_0\boldsymbol{\varepsilon}$ through decoder

Correlated VAE Implementation

- Architecture: Encoder outputs $K + K(K+1)/2$ nodes
  - K nodes for $\boldsymbol{\mu}_0$, and $K(K+1)/2$ nodes for L_0 lower triangular
- Sampling: Calculate $G_0 = e^{L_0}$
  - Note G_0 is lower triangular, nonsingular
  - Send sample $\boldsymbol{z} = \boldsymbol{\mu}_0 + G_0 \boldsymbol{\varepsilon}$ through decoder
- KL Divergence: Calculate $\Sigma_0 = G_0 G_0^T$

Correlated VAE Implementation

- Architecture: Encoder outputs $K + K(K+1)/2$ nodes
  - K nodes for $\boldsymbol{\mu}_0$, and $K(K+1)/2$ nodes for L_0 lower triangular
- Sampling: Calculate $G_0 = e^{L_0}$
  - Note G_0 is lower triangular, nonsingular
  - Send sample $\boldsymbol{z} = \boldsymbol{\mu}_0 + G_0 \boldsymbol{\varepsilon}$ through decoder
- KL Divergence: Calculate $\Sigma_0 = G_0 G_0^T$
  - Claim: Σ_0 is has positive determinant and is symmetric positive definite

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

G_0 is lower triangular, since addition and multiplication preserve lower triangular. G_0 is also nonsingular:

$$\det G_0 = \det e^{L_0} = e^{\operatorname{tr} L_0} \neq 0$$

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

G_0 is lower triangular, since addition and multiplication preserve lower triangular. G_0 is also nonsingular:

$$\det G_0 = \det e^{L_0} = e^{\operatorname{tr} L_0} \neq 0$$

Set $\Sigma_0 := G_0 G_0^T$. Now for any nonzero $\boldsymbol{y} \in \mathbb{R}^k$,

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

G_0 is lower triangular, since addition and multiplication preserve lower triangular. G_0 is also nonsingular:

$$\det G_0 = \det e^{L_0} = e^{\operatorname{tr} L_0} \neq 0$$

Set $\Sigma_0 := G_0 G_0^T$. Now for any nonzero $\boldsymbol{y} \in \mathbb{R}^k$,

$$\langle \Sigma_0 \boldsymbol{y}, \boldsymbol{y} \rangle = \boldsymbol{y}^T \Sigma_0 \boldsymbol{y} = \boldsymbol{y}^T G_0 G_0^T \boldsymbol{y} = \langle G_0^T \boldsymbol{y}, G_0^T \boldsymbol{y} \rangle = ||G_0^T \boldsymbol{y}||_2^2 > 0$$

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

G_0 is lower triangular, since addition and multiplication preserve lower triangular. G_0 is also nonsingular:

$$\det G_0 = \det e^{L_0} = e^{\operatorname{tr} L_0} \neq 0$$

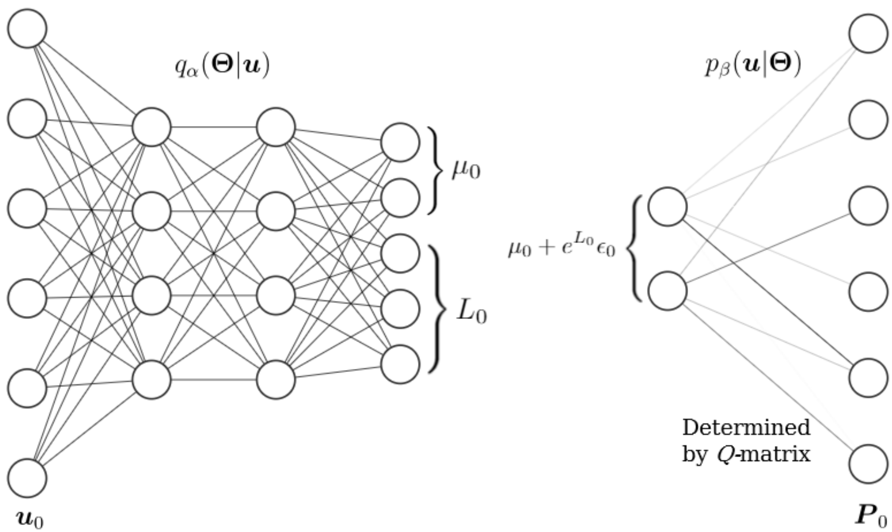
Set $\Sigma_0 := G_0 G_0^T$. Now for any nonzero $\boldsymbol{y} \in \mathbb{R}^k$,

$$\langle \Sigma_0 \boldsymbol{y}, \boldsymbol{y} \rangle = \boldsymbol{y}^T \Sigma_0 \boldsymbol{y} = \boldsymbol{y}^T G_0 G_0^T \boldsymbol{y} = \langle G_0^T \boldsymbol{y}, G_0^T \boldsymbol{y} \rangle = || G_0^T \boldsymbol{y} ||_2^2 > 0$$

Further,

$$\det \Sigma_0 = \det \left(G_0 G_0^\top \right) = \det G_0 \cdot \det G_0^\top = e^{\operatorname{tr} L_0} \cdot e^{\operatorname{tr} L_0} > 0$$

VAE architecture for correlated latent traits



Encoder structure for VAE learning $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$

Comparison of ML2P-VAE vs Other Methods

- MH-RM
- MC-EM
- QMC-EM
- ML2P-VAE$_{full}$
  - Assume full knowledge of correlation matrix Σ_1
  - Fit VAE with $\mathcal{N}(\mathbf{0}, \Sigma_1)$
- ML2P-VAE$_{est}$
  - Unknown correlation matrix $\Sigma_1 \Rightarrow$ estimate it with $\tilde{\Sigma}_1$
  - Fit VAE with $\mathcal{N}(\mathbf{0}, \tilde{\Sigma}_1)$
- ML2P-VAE$_{ind}$
  - Unknown correlation matrix $\Sigma_1 \Rightarrow$ assume independent $\mathbf{\Theta}$
  - Fit VAE with $\mathcal{N}(\mathbf{0}, I)$

Datasets

	Items	Skills	Students
ECPE	28	3	2,922
Sim-6	50	6	20,000
Sim-20	200	20	50,000
Sim-4	27	4	3,000

Sim-6 Discrimination Parameter Estimates

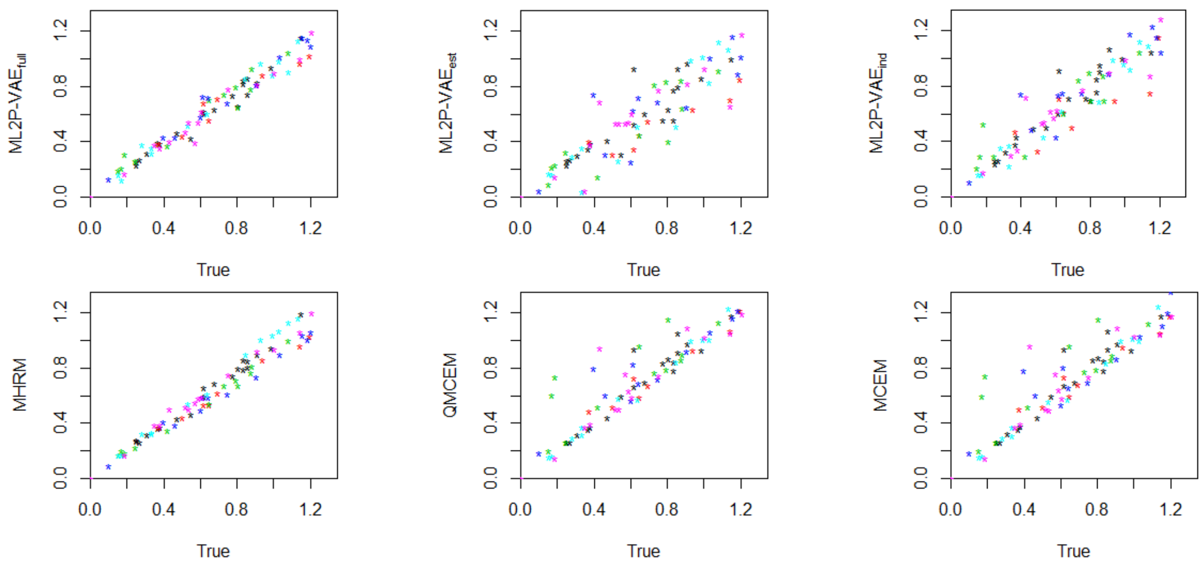


Figure 1: Correlation plots of discrimination parameter estimates for the Sim-6 dataset with 50 items and 6 latent traits. ML2P-VAE estimates are on the top row, and traditional method estimates are on the bottom row.

ECPE Parameter Estimates

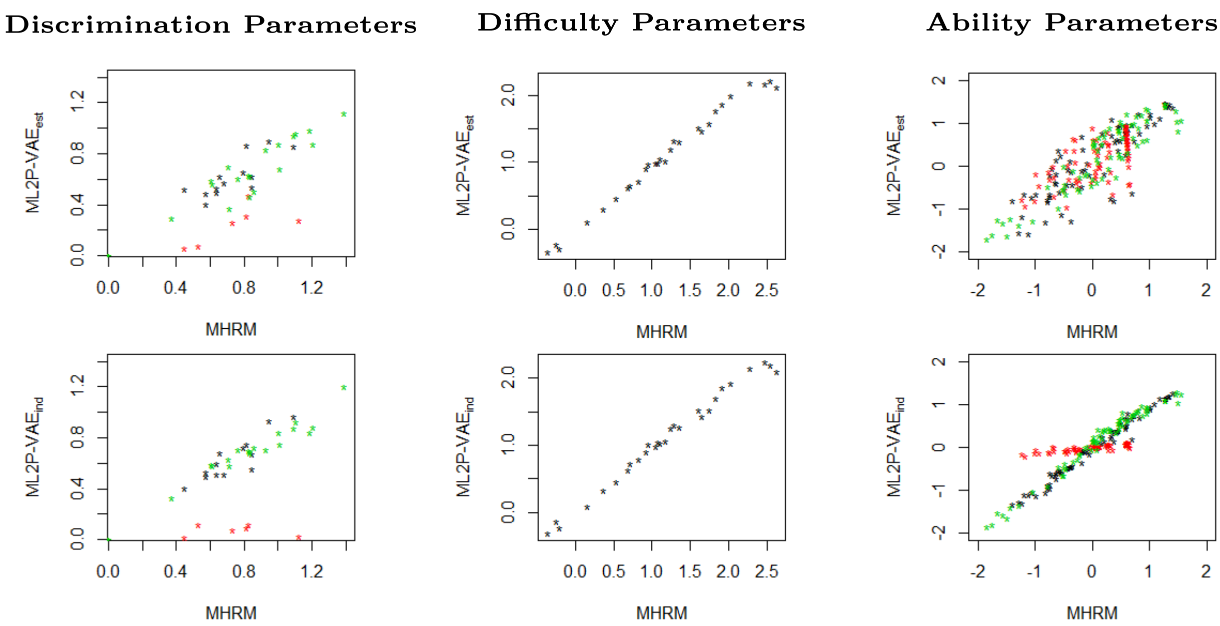


Figure 2: Estimates from ML2P-VAE methods plotted against "accepted" MHRM estimates from the ECPE dataset.

Sim-20 Parameter Estimates

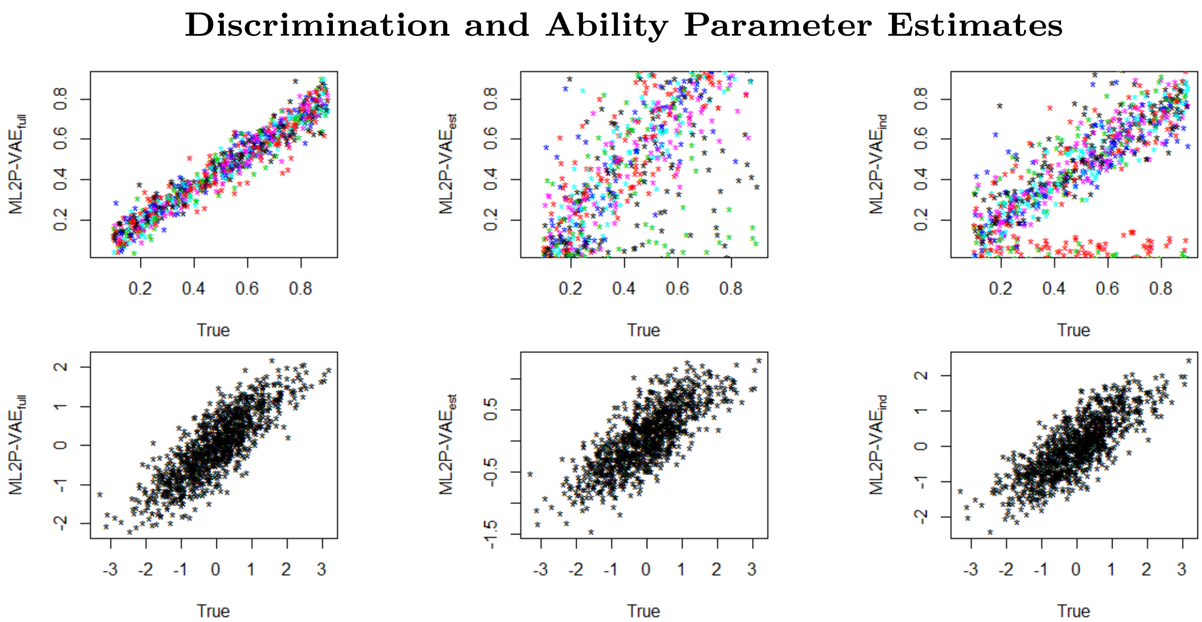


Figure 3: ML2P-VAE parameter estimates for Sim-20 with 200 items and 20 latent traits. The top row shows discrimination parameter correlation, and the bottom row shows ability parameter correlation.

Sim-4 Discrimination Parameter Estimates

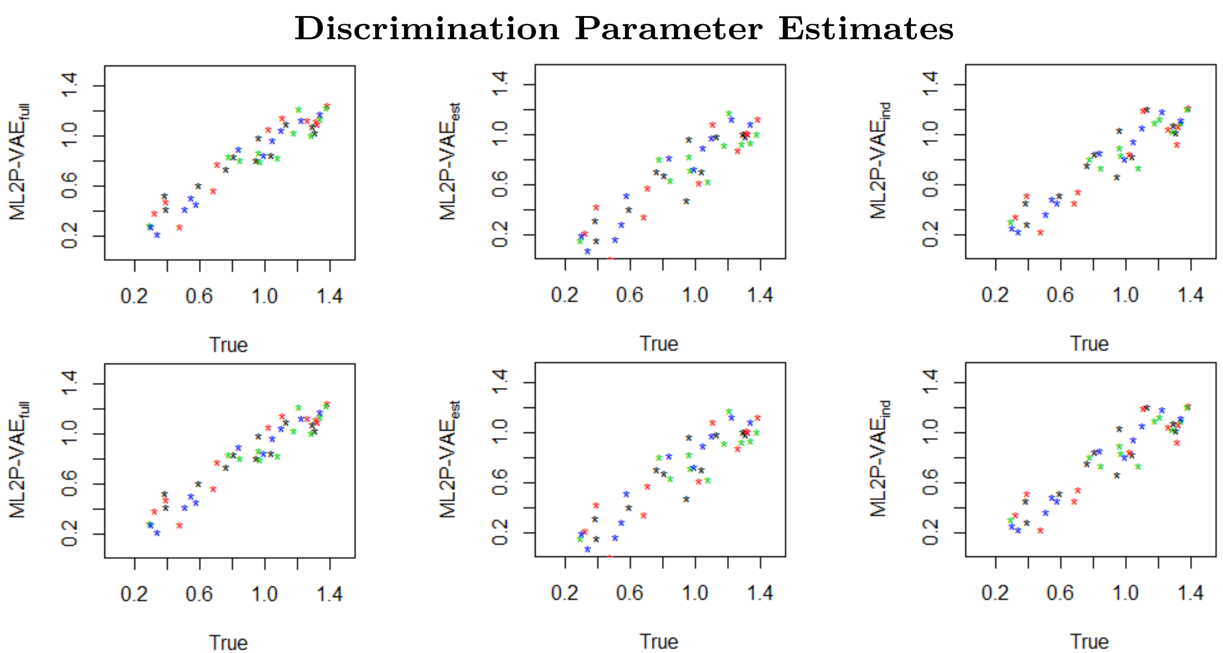


Figure 4: Discrimination parameter estimates for Sim-4 with 27 items and 4 latent skills. The top row shows estimates from ML2P-VAE methods, and the bottom row gives estimates yielded by traditional methods.

Correlated ML2P-VAE Results

Data Set	Method	$\boldsymbol{a}$.RMSE	$\boldsymbol{a}$.BIAS	$\boldsymbol{a}$.COR	$\boldsymbol{b}$.RMSE	$\boldsymbol{b}$.BIAS	$\boldsymbol{b}$.COR	$\boldsymbol{\Theta}$.R
(i) 6 abilities Sim-6	MHRM	0.0693	0.0319	0.9986	0.0256	-0.0021	0.9999	0.
	QMCEM	0.149	-0.067	0.9939	0.0376	-0.002	0.9998	0.7
	MCEM	0.1497	-0.0633	0.9936	0.0383	0.0035	0.9997	0.7
	ML2P-VAE$_{full}$	0.0705	0.0255	0.9985	0.0471	-0.0079	0.9996	0.
	ML2P-VAE$_{est}$	0.1803	0.0871	0.9891	0.064	-0.0131	0.9993	0.7
	ML2P-VAE$_{ind}$	0.1218	-0.0004	0.9944	0.0597	-0.0145	0.9994	0.7
(ii) 3 abilities ECPE	MHRM*	0*	0*	1*	0*	0*	1*	
	QMCEM	0.0159	0.0035	0.9999	0.0067	-0.0005	1	0.
	MCEM	0.0228	0.0148	0.9998	0.0064	-0.0008	1	0.
	ML2P-VAE$_{full}$	N/A	N/A	N/A	N/A	N/A	N/A	N
	ML2P-VAE$_{est}$	0.2794	0.2152	0.9713	0.148	0.0951	0.993	0.
	ML2P-VAE$_{ind}$	0.3208	0.2184	0.9504	0.154	0.0872	0.9932	0.3
(iii) 20 abilities Sim-20	MHRM	N/A	N/A	N/A	N/A	N/A	N/A	N
	QMCEM	N/A	N/A	N/A	N/A	N/A	N/A	N
	MCEM	N/A	N/A	N/A	N/A	N/A	N/A	N
	ML2P-VAE$_{full}$	0.078	0.0473	0.9983	0.0608	0.0054	0.9996	0.
	ML2P-VAE$_{est}$	0.2992	-0.1304	0.9822	0.1655	0.1215	0.9987	0.7
	ML2P-VAE$_{ind}$	0.2043	0.0592	0.9792	0.0958	-0.0029	0.9992	0.7
(iv) 4 abilities Sim-4	MHRM	0.0953	-0.0158	0.9966	0.0614	-0.0101	0.9988	0.
	QMCEM	0.0938	-0.0160	0.9967	0.0614	-0.0179	0.9989	0.
	MCEM	0.0951	-0.0138	0.9966	0.0644	-0.0199	0.9987	0.
	ML2P-VAE$_{full}$	0.1326	0.0780	0.9960	0.0872	-0.0311	0.9978	0.
	ML2P-VAE$_{est}$	0.2526	0.2106	0.9883	0.1035	-0.0337	0.9980	0.
	ML2P-VAE$_{ind}$	0.1658	0.1099	0.9939	0.0944	-0.0254	0.9976	0.

Scalability of ML2P-VAE

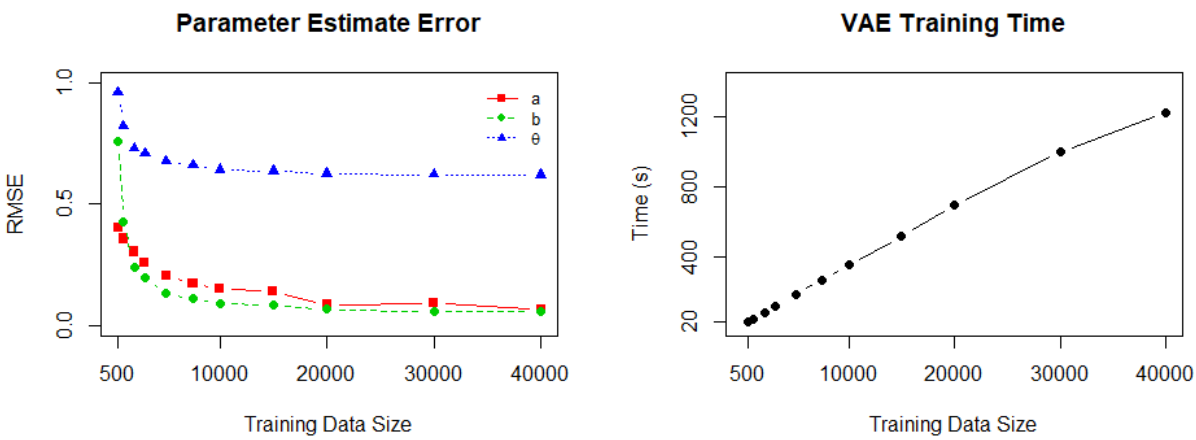


Figure 5: Performance of ML2P-VAE$_{full}$ on data set (iii) when trained on data sets of increasing size. The left plot gives the test RMSE after using different sizes of training data, and the right plot shows the time required to train the neural network.

ML2Pvae Package in R

- Software package on CRAN for easy implementation of ML2P-VAE methods
    - For IRT researchers – requires no knowledge of neural networks or TensorFlow

ML2Pvae Package in R

- Software package on CRAN for easy implementation of ML2P-VAE methods
  - For IRT researchers – requires no knowledge of neural networks or TensorFlow
- Package functions:
  - Construct ML2P-VAE model to desired architecture
    - Option for independent latent traits or full covariance matrix
  - Wrapper function to train neural network
  - Simple functions to obtain parameter estimates after training

Bayesian Knowledge Tracing

TODO: background/motivation of KT

RNN / LSTM

RNN, LSTM

Attention-based networks

Transformer / Attention

Deep Knowledge Tracing

DKT

SAKT

SAKT

Other methods

might want to mention DKVMN or PFA

Do deep models actually “trace” knowledge?

motivation

IRT-inspired Knowledge Tracing

method description

IRT-inspired Knowledge Tracing

image of architecture

Datasets

datasets

Results

Table and theta trace plot

Recovery of IRT parameters

disc and theta recovery plots

Learning a Q-matrix

cor heatmap and clustering

Extending ML2P-VAE to other IRT models

3PL and Samejima

Other application areas

BDI and personality questionnaires

Utilizing more domain knowledge

use Q matrix in attn calculation missing responses with embedding of interactions

Summary

Summary

Thank you!

References

TODO: choose citations in the right way

[1] Wainer and Thissen, D. “Test Scoring”. Erlbaum Associates, Publishers, 2001.

[2] da Silva, Liu, Huggins-Manley, Bazan. “Incorporating the Q-matrix into Multidimensional Item Response Models.” Journal of Educational and Psychological Measurement, 2018.

[3] Baker and Kim. “Item Response Theory: Parameter Estimation Techniques”. CRC Press, 2004.

[4] Bock and Aitken. “Marginal Maximum Likelihood Estimation of Item Parameters: Application of an EM Algorithm”. Psychometrika, 1981.

[5] Nielsen, Michael. “Neural Networks and Deep Learning”. Determination Press, 2015.

[6] Curi, Converse, Hajewski, Oliveira. “Interpretable Variational Autoencoders for Cognitive Models.” In Proceddings of the International Joint Conference on Neural Networks (IJCNN), 2019.

[7] 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.

[8] Converse, Curi, Oliveira. “Autoencoders for Educational Assessment.” In Proceedings of the Conference on Artifical Intelligence in Education (AIED), 2019.

[9] Converse, Curi, Oliveira, and Arnold. “Variational Autoencoders for Baseball Player Evaluation.” In Proceedings of the Fuzzy Systems and Data Mining Conference (FSDM), 2019.

Neural Network Methods for Application in Educational Measurement

Geoffrey Converse

University of Iowa

July 15, 2021